



Facultad de Ciencias

**Minería de flujos de datos para mantenimiento
predictivo en entornos industriales**

Data stream mining for predictive maintenance in
industrial environments

Trabajo de fin de máster
para acceder al

MÁSTER EN INGENIERÍA INFORMÁTICA

Autor: Ricardo Dintén Herrero

Directora: Marta Elena Zorrilla Pantaleón

Mayo 2021

Resumen

En los últimos años, la cuarta revolución industrial, también conocida como Industria 4.0, ha tomado una gran relevancia en el ámbito de la investigación. Esta nueva industria tiene como objetivo revolucionar la fabricación y producción gracias al uso de las nuevas tecnologías como la computación en la nube, la inteligencia artificial y los avances tecnológicos en la maquinaria de producción, que cada día está equipada con más sensores y con un hardware más potente. En el grupo de investigación ISTR se está desarrollando RAI4.0, una arquitectura de referencia apoyada en tecnologías Big Data para dar soporte a sistemas industriales de cuarta generación. Dentro de este proyecto se han implementado casos de uso relacionados con el procesado y almacenamiento de flujos de datos, sin embargo, no se han aplicado técnicas de análisis de datos ni inteligencia artificial, que son habilitadores clave de la I4.0. Por ello el objeto de este trabajo ha sido el desarrollo de un proceso de minería de datos sobre un flujo continuo de datos de carácter industrial. Para demostrar su aplicabilidad en la I4.0 se ha implementado un sistema de mantenimiento predictivo, para la supervisión de una bomba de agua que da suministro a un pueblo y se ha desplegado la solución en la nube de Amazon Web Services. De la experiencia extraída durante el desarrollo del proyecto se han identificado algunas dificultades que presenta el análisis de flujos de datos en entornos big data, así como señalado algunos retos pendientes dentro de este ámbito.

Palabras clave: Mantenimiento predictivo, Minería de flujo de datos, big data, industria 4.0

Abstract

In recent years, the fourth industrial revolution, also known as Industry 4.0, has taken on great relevance in the field of research. This new industry aims to revolutionise manufacturing and production thanks to the use of new technologies such as cloud computing, artificial intelligence and technological advances in production machinery, which is increasingly equipped with more sensors and more powerful hardware. The ISTR research group is developing RAI4.0, a reference architecture based on Big Data technologies to support fourth-generation industrial digital platforms. Different use cases related to the processing and storage of data streams have already been implemented in this project; however, none has been carried out using data analysis and artificial intelligence techniques, which are key enablers of I4.0. Therefore, the aim of this work has been the development of a data mining process on an industrial data stream. To demonstrate its applicability to I4.0, a predictive maintenance system has been implemented in order to monitor the behavior of a water pump. The solution has been deployed in the Amazon Web Services cloud. From the experience gained during the development of the project, some issues in analyzing data streams in big data environments have been identified, as well as some research challenges still opened in this field.

Key words: predictive maintenance, data stream mining, big data, industry 4.0

Índice

Índice de figuras	v
Índice de tablas	vii
Índice de <i>scripts</i>	ix
1. Introducción	1
2. Mantenimiento predictivo	3
2.1. Introducción	3
2.2. Minería de datos	4
2.2.1. Aprendizaje automático	4
2.2.2. Técnicas de modelado	5
2.2.3. Técnicas de muestreo	7
2.2.4. Técnicas de tratamiento de valores nulos	7
2.2.5. Métricas empleadas para evaluar modelos	8
2.2.6. Técnicas para el entrenamiento, validación y ajuste de modelos . . .	9
2.2.7. Minería de flujos continuos de datos	9
2.3. Trabajos relacionados	9
3. Arquitectura y tecnologías	13
3.1. Arquitectura de referencia RAI4.0	13
3.2. Tecnologías empleadas	15
3.2.1. Lenguajes de programación y frameworks de desarrollo	15
3.2.2. Tecnologías big data	16
3.2.3. Librerías de machine learning & data science.	17
3.2.4. Infraestructura	18
4. Metodología de trabajo	21
4.1. Metodología de trabajo del proyecto y planificación general	21
4.2. Metodologías para data minería de flujos de datos	22
4.2.1. KDD	23
4.2.2. SEMMA	23
4.2.3. CRISP-DM	24
5. Caso de estudio	27

5.1.	FASE 1: Entendimiento del problema	27
5.2.	FASE 2: Entendimiento del conjunto de datos	27
5.3.	FASE 3: Preparación de los datos	30
5.4.	FASE 4: Modelado	33
5.5.	FASE 5: Evaluación	33
5.6.	FASE 6: Despliegue	34
5.6.1.	Infraestructura	34
5.6.2.	Bus de datos	39
5.6.3.	Servicio de planificación	39
5.6.4.	Ejecución del sistema	43
5.7.	Interfaz de usuario	44
5.7.1.	WebSocket	44
5.7.2.	Capa de presentación	45
6.	Aportaciones metodológicas y consideraciones para el análisis de flujos de datos	47
7.	Conclusiones y líneas futuras	49
8.	Referencias	51

Índice de figuras

1.	Plataforma como servicio basado en la arquitectura de referencia RAI4.0 . . .	14
2.	Organización de tecnologías en el sistema de mantenimiento predictivo . . .	15
3.	Diagrama gantt del proyecto de minería de flujos de datos	22
4.	Vista de las 5 primeras filas del conjunto de datos	28
5.	Descripción estadística de las variables del conjunto de datos	28
6.	Evolución temporal de los valores enviados por diferentes sensores y el estado de la máquina	28
7.	Correlación entre variables representada mediante mapa de calor	29
8.	Porcentaje de instancias del conjunto de datos que hay para cada estado de la bomba	30
9.	Proporción de filas que contienen algún valor nulo	31
10.	Pipelines de preprocesamiento empleadas para la construcción de los modelos predictivos	32
11.	Interfaz de usuario de Spark con los workers y los recursos disponibles . . .	40
12.	Interfaz de usuario de Spark con el workflow desplegado	42
13.	Ejecución del sistema por consola	43
14.	Comportamiento temporal del workflow en la interfaz de usuario de Spark	44
15.	Interfaz de usuario desarrollada con Angular	45
16.	Clasificación de retos del análisis de Big Data[1]	48

Índice de tablas

1.	Matriz de confusión	8
2.	Evaluación de los predictores contruidos para los diferentes conjuntos de datos y algoritmos seleccionados	33
3.	Tiempo(ms) requerido por cada modelo para realizar una predicción	34

Índice de *scripts*

5.1. Fichero de configuración del inventario dinámico	35
5.2. Script Ansible para aprovisionar las máquinas virtuales de Amazon Web Services	35
5.3. Script Ansible para preparar el entorno de trabajo	37
5.4. Script Ansible para liberar todos los recursos aprovisionados mediante Ansible	38
5.5. Script para la conexión con la fuente de datos para iniciar el stream	40
5.6. Script encargado del procesamiento del stream para obtener las lecturas de bus de datos	40
5.7. Función para realizar predicciones sobre cada fila de datos	41
5.8. Definición del sumidero y comienzo del streaming	42
5.9. Comando para iniciar la ejecución de Spark	42
5.10. Comando para iniciar la ejecución del stream-ingestor y enviar los datos a kafka	43

1 Introducción

Este trabajo surge dentro del marco del proyecto del plan Nacional Sistemas Informáticos Predecibles y Confiables para la Industria 4.0 (TIN2017- 86520-C3-3 R) en el que participa el grupo de Ingeniería de Software y Tiempo Real de la Universidad de Cantabria. En este proyecto se está desarrollando una arquitectura de referencia que guíe en el diseño e implementación de aplicaciones reactivas que manejen en tiempo real la información generada en las plantas de producción de cuarta generación.

En primer lugar, se debe explicar qué representa el concepto Industria 4.0. Este concepto surge en Alemania en el año 2013 [2] para representar la integración de la computación distribuida y el aumento de las capacidades de los sistemas ciberfísicos y embebidos en las actividades de producción y logística así como en el uso de internet en los procesos industriales. Se trata de nueva revolución industrial, como en su día fueron la máquina de vapor, la producción en cadena y la energía eléctrica así como la automatización y las tecnologías de la información.

En la actualidad, la inteligencia artificial (IA) y la minería de datos (MD) están en auge debido a los avances en la capacidad de cómputo de los sistemas y de la cantidad ingente de datos que se generan y recogen cada día. Los sitios web de ventas o las redes sociales hacen uso de estas tecnologías desde hace años con el objeto de mejorar su tasa de conversión y su alcance, lo cual hace que tanto su servicio como sus ganancias mejoren notablemente. En el entorno industrial, debido a las restricciones temporales y a estándares de calidad mucho más estrictos, sumado a una maquinaria que cuenta en su gran mayoría con sistemas informáticos específicos o embebidos, no ha tenido tanta popularidad. Sin embargo, los nuevos sistemas ciberfísicos y de sensorización, cada vez más capaces, pueden hacer que las plantas de producción puedan aprovechar los beneficios de la inteligencia artificial y la minería de datos y optimizar sus procesos mejorando entre otros la productividad, la eficiencia y la reducción de costes.

En este trabajo se va a implementar un sistema de mantenimiento predictivo basado en la minería de flujos de datos (en inglés conocido como data stream mining), desplegado sobre una plataforma de tercera generación, que cumpla con los requisitos establecidos en la arquitectura de referencia RAI4.0[3], haciendo uso de tecnologías big data. Teniendo en cuenta el marco de referencia en el que se realiza se pretenden alcanzar los siguientes objetivos:

- Analizar el estado del arte en el ámbito del mantenimiento predictivo y la minería de flujos de datos continuos.
- Seleccionar el software IA adecuado a nuestro propósito y desplegar una plataforma de tercera generación siguiendo las directrices marcadas por la arquitectura de referencia RAI4.0[3] sobre la que instanciar un sistema de mantenimiento predictivo.
- Implementar un caso de uso de mantenimiento predictivo siguiendo la metodología de trabajo específica CRISP-DM adaptada al contexto big data y analizar los

resultados.

- Extraer conclusiones metodológicas para la implantación de sistemas de mantenimiento predictivo en plataformas de tercera generación

El documento se ha estructurado del siguiente modo: el capítulo 2 introduce el campo de la minería de datos y brevemente resume trabajos y proyectos relacionados con el mantenimiento predictivo encontrados en la literatura. En el capítulo 3, se presentan la arquitectura y las tecnologías empleadas en el proyecto para construir y desplegar el sistema de mantenimiento predictivo. En el capítulo 4, se expone un diagrama Gantt con las tareas y el tiempo dedicado a cada una en el desarrollo de este TFM; asimismo se describe la metodología seguida en el proyecto y se extiende para su aplicación a stream mining. En el capítulo 5, se describe el proceso de construcción y despliegue de un modelo predictivo siguiendo la metodología CRISP-DM. En el capítulo 6, se exponen recomendaciones para el análisis big data y se identifican retos pendientes en el ámbito. Por último, en el capítulo 7, se presentan las conclusiones extraídas de la realización del proyecto y se enuncian posibles líneas de trabajo futuro.

2 Mantenimiento predictivo

En ese capítulo se contextualiza el ámbito del trabajo y se describe someramente el campo de la minería de datos y trabajos relacionados en su aplicación al mantenimiento predictivo.

2.1. Introducción

Antes de comenzar este trabajo, conviene realizar un repaso al estado del arte en materia de mantenimiento predictivo. Por tanto, se va a comenzar por definir qué se entiende por mantenimiento predictivo y qué requisitos son necesarios para poder llevar a cabo mantenimiento predictivo dentro de un entorno industrial. A continuación, se resumen las propuestas de distintos autores para desarrollar sistemas de mantenimiento predictivo.

El mantenimiento predictivo consiste en utilizar técnicas de minería de datos e inteligencia artificial para predecir fallos o funcionamientos anómalos de una máquina antes de que sucedan y poder, de esta manera, planificar las tareas de mantenimiento y reparación. Para conseguir este objetivo, es necesario contar con una serie de sensores capaces de medir diferentes parámetros durante el funcionamiento del sistema en cuestión. Estas medidas serán las utilizadas para construir y mantener los modelos predictivos, así como para obtener las predicciones.

Para poder desarrollar una solución de mantenimiento predictivo no solo es necesario tener una serie de sensores que recojan valores, también es necesario tener algún protocolo de actuación frente a incidencias y un historial en el que se recojan todos los valores medidos por los sensores junto con el estado global del sistema que se está monitorizando. Esta información generalmente se dispone en los entornos industriales gracias a los métodos de mantenimiento correctivo y preventivo que tienen implementados. El primero consiste en realizar tareas de mantenimiento solamente cuando el sistema falla, parándose el proceso productivo mientras se realiza. El segundo consiste en realizar mantenimientos periódicos para revisar y reparar la maquinaria. En este caso, puede ocurrir que el sistema falle antes del mantenimiento programado, sin ninguna forma de detectarlo. El mantenimiento preventivo es el paso inmediatamente anterior al mantenimiento predictivo.

Ambos tipos de mantenimiento: correctivo y preventivo; tienen en común que se establece un protocolo de atención a las averías y se realiza un registro de todas las reparaciones, lo que es clave para poder comenzar a implementar una solución de mantenimiento predictivo.

2.2. Minería de datos

Una de las premisas del mantenimiento predictivo es el uso de técnicas de minería de datos y aprendizaje automático para ayudar a mantener la integridad y el buen funcionamiento de la maquinaria. A continuación, se introduce el concepto de minería de datos, algunos de los algoritmos más populares y las métricas empleadas para medir la efectividad de las soluciones. Asimismo, se introduce el concepto de minería de flujos de datos.

La minería de datos es una disciplina del campo de la estadística y la computación que tiene como objetivo extraer patrones desconocidos de grandes conjuntos de datos. Para ello, se apoya en técnicas de aprendizaje automático, inteligencia artificial y tecnologías de bases de datos.

2.2.1. Aprendizaje automático

Una parte muy importante de la minería de datos es el aprendizaje automático, que tiene por objeto conseguir que las máquinas extraigan conocimiento de manera automática a partir de un conjunto de datos. A continuación, se explican los tipos de algoritmo más empleados, en función de si se encuentran dentro del paradigma supervisado o no supervisado. El aprendizaje supervisado es aquel en el que contamos con una variable objetivo conocida que queremos predecir. Para este tipo de técnicas necesitamos un conjunto de datos en el que conozcamos tanto las variables que queremos analizar como la variable objetivo, de manera que los algoritmos de aprendizaje automático puedan establecer relaciones entre las diferentes variables en un proceso que se conoce como entrenamiento. Las técnicas más conocidas dentro del aprendizaje supervisado son:

- Clasificación: consiste en asignar cada instancia de datos a su clase, por ejemplo, identificar la especie de un animal que aparece en una foto. Puede ser clasificación binaria, si la variable objetivo se divide en 2 clases o multi-clase si la variable objetivo está compuesta de más de dos clases.
- Regresión: consiste en predecir un número en función de una serie de valores, por ejemplo, realizar una predicción de la probabilidad de lluvia en función de algunos parámetros meteorológicos como de la presión atmosférica, la temperatura y la humedad del ambiente.

El aprendizaje no supervisado es aquel en el que intentamos extraer patrones, reglas o agrupar las instancias de nuestro conjunto de datos. Para este tipo de aprendizaje no necesitamos contar con una variable objetivo ni con datos etiquetados para el proceso de entrenamiento. Las técnicas más comunes dentro del aprendizaje no supervisado son:

- Clustering: consiste en identificar y dividir los datos de un conjunto de datos en grupos con características individuales.
- Asociación: trata de identificar relaciones entre diferentes variables para encontrar reglas que permiten entender las causas de un evento, un ejemplo sencillo de este tipo de técnicas, es el análisis de las compras de un supermercado para encontrar productos que se suelen comprar juntos.

2.2.2. Técnicas de modelado

Para aplicar las diferentes técnicas se emplean diferentes modelos o algoritmos. En este apartado se repasan algunos de los más conocidos, explicando brevemente su funcionamiento y algunas ventajas e inconvenientes de cada uno de ellos.

Random Forest Classifier

Este tipo de clasificador consiste en la construcción de una gran cantidad de árboles de decisión que operan como un conjunto. Cada árbol ofrece una predicción de clase y la clase con más votos se convierte en la predicción del modelo. De manera que cada uno de los árboles modeliza una serie de características diferentes. Los bosques de decisión aleatorios corrigen el problema de los árboles de decisión de sobreajustarse a su conjunto de entrenamiento consiguiendo un modelo más preciso y robusto.

Ventajas

- Es una solución eficiente
- Consigue resultados bastante precisos

Inconvenientes

- Funciona mejor con conjuntos grandes de datos, por lo que es posible que si no se cuenta con un conjunto suficientemente grande no se consigan los resultados esperados.
- Es sensible al ruido (datos espúreos, faltantes y fuera de rango).

Support Vector Classifier

Este tipo de clasificador se basa en encontrar un conjunto de hiperplanos de dimensión $N-1$ donde N es el número de características recibidas por el modelo, de manera que pueda conseguir la mejor separación de los datos en las diferentes categorías. Para ello, representa cada instancia en un espacio de dimensión N y utiliza la distancia entre puntos para calcular el hiperplano con menor distancia a los diferentes conjuntos de datos.

Ventajas

- Suele conseguir buenos resultados con pocos datos
- Consigue resultados bastante precisos

Inconvenientes

- Coste computacional muy alto, tanto a nivel de procesamiento como de memoria
- Tiempos de entrenamiento muy altos

KNeighbors Classifier

Es un método de clasificación no paramétrica. Consiste en la construcción de k vectores característicos y la clase predicha será aquella cuya distancia es menor. En clasificación el resultado se obtiene mediante la clase de los k vecinos más cercanos. Si $k=1$ se le asigna directamente la clase de su vecino. Si $k>1$ se realiza una votación entre los vecinos más cercanos, y finalmente se asigna la clase más común entre sus vecinos. Al basarse en el cálculo de distancias, este algoritmo requiere que las variables de entrada estén normalizadas.

Ventajas

- No requiere una fase de entrenamiento, esto es de ajuste de parámetros. Se trata de una técnica descriptiva en el que a cada punto se le asigna una etiqueta.
- Es un método bastante sencillo y eficiente.

Inconvenientes

- Al tener en cuenta las distancias para escoger los vecinos, es necesario estandarizar los datos.
- Cuando el número de vecinos es alto se consigue un algoritmo más robusto, pero si la muestra está sesgada es posible que no generalice correctamente.

Redes Neuronales Artificiales

Las redes neuronales artificiales son un tipo de modelo computacional inspirado en las redes neuronales presentes en el cerebro. Están formadas por un grupo de neuronas artificiales conectadas entre sí. El funcionamiento se basa en recibir un valor que se va propagando por las diferentes neuronas y transformándose mediante una serie de funciones que pueden ser lineales o no lineales para obtener un valor nuevo a la salida de la red. Cada neurona tiene una función de activación y cada enlace (denominado sinápsis) tiene un peso, de esta forma cuando una neurona recibe un valor, aplica la función y el valor resultante se multiplica por el peso del enlace con la siguiente neurona. El aprendizaje se consigue mediante un correcto ajuste de los pesos en los enlaces.

Ventajas

- Son capaces de encontrar patrones y relaciones muy complejas
- Consigue resultados bastante precisos

Inconvenientes

- Coste computacional muy alto, tanto a nivel de procesamiento como de memoria. Generalmente se precisa de un acelerador hardware para computación paralela como las GPU.
- Necesitan un tamaño de muestra muy elevado para alcanzar un buen rendimiento.

2.2.3. Técnicas de muestreo

A continuación se enumeran las técnicas más habituales para la selección de los conjuntos de datos de entrenamiento y prueba.

Muestreo aleatorio

Como el nombre indica, esta técnica de muestreo consiste en escoger un determinado número de elementos de manera aleatoria.

Muestreo aleatorio estratificado

Esta estrategia divide la muestra en grupos y realiza un muestreo aleatorio de cada grupo. De esta manera se respeta la proporción de valores de cada grupo que existía en el conjunto original.

Muestreo sistemático

Esta estrategia de muestreo consiste en establecer un intervalo fijo para la selección de los elementos de la muestra.

Muestreo por grupos

En esta estrategia se divide el conjunto de datos en varios grupos de igual tamaño y se escoge cada subconjunto siguiendo un intervalo fijo.

2.2.4. Técnicas de tratamiento de valores nulos

A continuación se describen brevemente una serie de técnicas que se utilizan habitualmente para resolver el problema de datos faltantes en los conjuntos de datos de entrenamiento.

Eliminar valores nulos

En este primer caso, la propuesta es eliminar aquellas instancias (filas de datos) que contengan valores nulos. Si el conjunto de datos es suficientemente grande y no tiene demasiados valores nulos, esta podría ser una estrategia rápida y efectiva para deshacerse de los valores nulos. Sin embargo, si el porcentaje de valores nulos es muy elevado puede conllevar a no disponer de información suficiente para realizar el entrenamiento.

Imputar valores

Esta estrategia consiste en sustituir los valores faltantes por otros. Generalmente se suelen sustituir los valores por la media de la variable si es numérica o por la moda si es una variable categórica. Esta aproximación hace que la media o la moda de las variables no se vea afectada. Sin embargo, cuando el número de variables de la muestra es elevada puede modificar la varianza.

Utilizar un predictor

Esta estrategia tiene como objetivo mitigar los efectos que tiene el método anterior en la varianza. Para ello se entrena un modelo de regresión que sea capaz de generar o estimar los valores que podrían tomar esas variables sin alterar la distribución. El problema de este método es que se generan valores de forma más o menos aleatoria que podría introducir ruido, por tanto, es una buena práctica introducir una nueva variable que identifique qué filas tienen valores reales y cuáles son generados.

2.2.5. Métricas empleadas para evaluar modelos

A continuación se definen las métricas más habituales para evaluar los predictores construidos. Estas se definen a partir de la matriz de confusión o matriz de error (ver Tabla 1), una tabla que describe el rendimiento de un modelo supervisado en los datos de prueba, donde:

		Predicción	
		+	-
Real	+	TP	FN
	-	FP	TN

Tabla 1: Matriz de confusión

- True Positives (TP): número de casos que el modelo predice correctamente la clase positiva.
- True Negatives (TN): número de casos que el modelo predice correctamente la clase negativa.
- False Positives (FP): número de casos en que el modelo predice como positivo siendo la clase negativa.
- False Negatives (FN): número de casos en que el modelo predice como negativo siendo la clase positiva.
- Accuracy: esta métrica representa el porcentaje de predicciones correctas respecto al total de predicciones realizadas. Es una buena medida cuando las clases están casi equilibradas. $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$
- Sensibilidad o TPR: es el porcentaje de casos positivos reales detectados respecto al total de casos positivos reales en la muestra. $Sensibilidad = \frac{TP}{TP+FN} = Recall$
- Especificidad o TNR: el porcentaje de casos negativos reales detectados respecto al total de casos negativos. $Especificidad = \frac{TN}{TN+FP}$
- Tasa de falsa alarma o tasa de falsos positivos: tasa de casos incorrectamente clasificados como positivos respecto del total de casos negativos en el conjunto. $FAR = \frac{FP}{FP+TN}$
- F-score: es una métrica que combina (media armónica) la precisión y el recall para comparar clasificadores. $f_{score} = \frac{TP}{TP + \frac{1}{2}(FP+FN)}$
- Curva ROC: es una representación de la sensibilidad frente a la especificidad. También se puede interpretar como el TPR frente al FAR.

2.2.6. Técnicas para el entrenamiento, validación y ajuste de modelos

- Validación cruzada: la validación cruzada es una técnica de entrenamiento y evaluación de modelos que permite obtener una evaluación de un estimador de manera más fiable y robusta. La técnica consiste en dividir la muestra de entrenamiento en un determinado número de subconjuntos disjuntos, apartar uno de esos conjuntos para validar el estimador y utilizar el resto para el entrenamiento. Este proceso se debe repetir tantas veces como subconjuntos se hayan creado, escogiendo cada vez uno diferente para la validación. Finalmente, se promedian los resultados obtenidos en cada una de las iteraciones. La variante más extendida es 10-fold. Existe una variante que se conoce como leave-one-out, en la que se divide el conjunto de entrenamiento en tantas partes como elementos contenga el conjunto original, que se utiliza cuando el conjunto de datos de entrenamiento es reducido.
- Grid Search: técnica para ajustar los parámetros de un modelo predictivo que consiste en definir un mapa con diferentes valores para cada parámetro y probar de manera automática todas las combinaciones posibles para encontrar y seleccionar la mejor para el conjunto de datos.

2.2.7. Minería de flujos continuos de datos

En los últimos años, y como consecuencia del big data, el IoT y la creciente sensorización de casi cualquier dispositivo electrónico, ha surgido una nueva corriente dentro de la minería de datos, la minería de flujos de datos. Esta trata de aplicar las técnicas de minería de datos a los flujos de datos como los generados por los sensores de los dispositivos electrónicos, o las interacciones de las personas en la web.

Según el entorno en el que se desarrolle el proceso de minería de datos el objetivo o las técnicas podrían ser diferentes. Por ejemplo, en un entorno industrial en el que se pretende monitorizar una serie de elementos mecánicos, se puede entrenar un predictor y aplicarlo para realizar predicciones sobre los valores recogidos por los sensores. Esto es así, porque la maquinaria generalmente tiene unas condiciones de trabajo concretas, y si estas cambian de forma rápida o brusca probablemente se debe a un fallo del sistema. Sin embargo, en otro entorno, como una tienda online, los datos pueden seguir modas o tendencias cambiantes, de manera que el predictor tiene que adaptarse a estos cambios y descartar los datos antiguos que solamente introducen ruido y errores en las predicciones. Este cambio de tendencia es lo que se conoce como *concept drift*. Para contrarrestar este efecto es necesario que los modelos se retroalimenten con los nuevos datos que van llegando, lo que supone adaptar los algoritmos existentes o crear unos nuevos capaces de detectar este cambio.[4]

2.3. Trabajos relacionados

A continuación, se relacionan trabajos encontrados en la literatura donde se han aplicado técnicas de inteligencia artificial a mantenimiento predictivo.

En Li et al. (2014) [5] llevan a cabo una propuesta de un sistema de mantenimiento predictivo para evitar descarrilamiento a partir de unos sensores instalados en las vías del tren. Los sensores empleados recogen datos de telemetría como pueden ser temperatura, impacto y fuerza lateral, así como el sonido producido por las ruedas. Para ser capaz

de gestionar el volumen de datos generado por una red de ferrocarriles que se encuentra en continuo crecimiento, como también lo hace el número de sensores instalados para realizar la monitorización de los sistemas, es necesario el uso de tecnologías big data. Con los datos de estos sensores trataron de construir dos sistemas de predicción: el primero, para tratar de predecir alarmas antes de que se generaran y el segundo, para detectar defectos en las ruedas y vagones del tren. Su propuesta en el primero de los casos fue emplear Máquinas Vector Soporte (SVM) y en el segundo, árboles de decisión. Para hacer la configuración de parámetros de los algoritmos aplicaron la técnica grid search con diferentes valores para encontrar los que arrojaban los mejores resultados. Para evaluar los modelos predictivos construidos utilizaron como criterio de evaluación TPR y FAR debido a que el conjunto de datos de entrenamiento no estaba equilibrado, presentaba más casos negativos que positivos. En el caso de los ferrocarriles, detener el convoy para realizar una inspección es un proceso muy costoso y, por tanto, se debe evitar. Este hecho, llevó a los autores a utilizar como métricas de evaluación True Positive Rate y False Alarm Rate, intentando maximizar el primero y minimizar el segundo. Los autores consiguieron unos valores realmente satisfactorios alcanzando un TPR del 97 % y un FAR del 0,23 %.

En Praveenkumar et al. (2014) [6] también emplearon SVM para desarrollar su modelo de mantenimiento predictivo. Pero en este caso el propósito del sistema era predecir fallos en la caja de cambios de un automóvil, empleado un sensor de vibraciones instalado junto a la caja de cambios. Para realizar el experimento montaron una caja de cambios de cuatro marchas en un laboratorio y fueron tomando medidas de las vibraciones primero con los engranajes en buen estado y después con engranajes defectuosos. Cada uno de los engranajes se probó bajo dos valores de Par (o Torque): 0 Kgm y 5Kgm; y dos velocidades diferentes: 500 y 750 rpm, lo que dio lugar a 32 escenarios diferentes. En este caso, a diferencia del anterior, se utilizó el accuracy como medida de evaluación ya que el conjunto de datos de entrenamiento estaba equilibrado (construido en el laboratorio). El experimento, consiguió unos buenos resultados, alcanzando precisiones comprendidas entre el 92 % (peor caso) y 100 % (mejor caso). Esto demuestra que la aplicación de técnicas de minería de datos son eficaces en tareas de detección y diagnóstico de averías en elementos mecánicos. Sin embargo, no realizaron una evaluación en un entorno real, donde la caja de cambios no se encuentra aislada en un laboratorio, sino que se encuentra rodeada de otros elementos mecánicos que producen vibraciones y podrían interferir o introducir ruido en los valores recogidos por los sensores.

En [7], Prytz et al. (2015) desarrollan una solución para estimar la vida útil restante de compresores instalados en autobuses y camiones. En concreto, se pretende detectar cuándo un componente no llegará hasta el próximo mantenimiento programado y poder adelantar o retrasar la sustitución del componente evitando así los sobrecostes de una avería en carretera. En el artículo se abordan principalmente dos problemáticas: el procesamiento de los datos y el horizonte temporal de las predicciones. El primero de los problemas viene dado porque al haber muchas muestras procedentes de cada vehículo, estas se encuentran fuertemente correlacionadas. Además, hay bastante ruido y muestras inconsistentes debido a problemas de cobertura durante las rutas de los vehículos y diferencias entre los parámetros disponibles en cada vehículo. Por tanto, los autores optan por limpiar el conjunto de datos de parámetros y vehículos con gran porcentaje de valores nulos. Otro problema que surge es que los datos no están equilibrados, por lo que se dispone de muchas más muestras de funcionamiento normal que de rotura del compresor, como también les ocurría a Li et al. (2014) en su trabajo. En este caso, los autores decidieron aplicar técnicas para aumentar el conjunto de datos y, así poder emplear muestras de mayor tamaño y no tan sesgadas por el número de instancias de cada tipo. Respecto al segundo de los problemas, el horizonte temporal, es un aspecto de suma importancia, ya que si el horizonte temporal

es pequeño las predicciones son más precisas, pero probablemente no suponen ninguna ventaja al no tener tiempo para reaccionar. Por el contrario, si el horizonte temporal es demasiado grande las predicciones pierden precisión, pero el margen de maniobra es mucho mayor en caso de que sea necesario replanificar el mantenimiento del vehículo. Por tanto, se plantean varios horizontes temporales para ver con cuál se obtiene un mejor compromiso entre precisión y utilidad de las predicciones. Para evaluar esto, los autores desarrollaron una métrica basada en el TPR, el FAR, los costes de mantenimiento y los costes de una interrupción no planificada del servicio. De esta forma consiguen estimar el beneficio que tendría para la empresa el emplear el sistema de mantenimiento predictivo y comparar los distintos modelos. Para las realizar las predicciones emplearon un modelo de tipo Random Forest, que debido a sus características funciona muy bien con un alto número de parámetros siendo bastante robusto y poco propenso al sobreajuste. En este caso, aplican PCA ¹ para reducir la dimensionalidad del conjunto de datos y comparan los resultados con los obtenidos empleando un conjunto de características seleccionadas por un experto. El resultado final es bastante satisfactorio en todos los aspectos ya que se alcanza una puntuación bastante buena en la métrica establecida por ellos, lo que significa que el ahorro potencial es bastante alto, concretamente entre 50€ y 120€ de ahorro medio por vehículo y revisión. Por último, cabe destacar que se obtienen unos resultados ligeramente mejores con la aproximación de selección de características dirigida por los datos, lo que nos muestra que estas técnicas son capaces de igualar y superar a los expertos en determinadas situaciones.

En [8], Canizo et al. (2017) crean una solución de mantenimiento predictivo para unas turbinas de aire instaladas en unos generadores. El objetivo de su trabajo es comparar el mantenimiento predictivo con el mantenimiento periódico utilizando un framework formado por diferentes tecnologías big data (Kafka, Spark, Zookeeper, Mesos, HDFS). Dividen el trabajo en dos partes o flujos de datos diferenciados; uno offline y otro online. El primero se refiere al almacenamiento de datos y entrenamiento de los modelos predictivos y el segundo a las predicciones sobre los datos nuevos. Al final del experimento, los autores señalan que consiguen alcanzar una alta sensibilidad y una baja especificidad con sus modelos.

Esta selección de trabajos muestra el interés y la aplicabilidad de la minería de datos al mantenimiento predictivo. Esto se debe a que la maquinaria suele tener una serie de condiciones y parámetros de funcionamiento bastante acotados, y esos modelos son capaces de detectar cambios en el funcionamiento con gran rapidez y precisión. Incluso, en el caso de Canizo et al.[8] se ha aplicado a un flujo de datos utilizando tecnologías big data.

¹Principal Component Analysis

3 Arquitectura y tecnologías

3.1. Arquitectura de referencia RAI4.0

Para desplegar el sistema de mantenimiento predictivo es necesario apoyarse en una plataforma que permita gestionar y manipular flujos de datos. En este caso, se ha tomado como base la plataforma propuesta por el grupo de ingeniería de software y tiempo real de la Universidad de Cantabria, conforme a su arquitectura de referencia.[3]

La arquitectura de referencia está diseñada para ofrecer soluciones big data y desarrollar aplicaciones distribuidas dentro del entorno industrial, donde las restricciones de tiempo y seguridad suelen ser un factor muy importante. Para cumplir los requisitos de los sistemas industriales de cuarta generación, la arquitectura RAI4.0[3] centrada en dato se basa en 3 principios clave:

- DaaS (Data as a Service): El entorno industrial se define al más alto nivel en base a la descripción y caracterización de la información que gestiona. Los componentes software y los recursos hardware que constituyen la plataforma son sólo medios que se reclutan y escalan a fin de que generen, procesen, den soporte o consuman los datos gestionados.
- PaaS (Platform as a Service): En el entorno industrial los recursos de cómputo que conforman la plataforma son de naturaleza heterogénea y cambian en el tiempo, ya sea por fallo de los componentes o por necesidades de escalar en función de la demanda. Por ello, en esta arquitectura se define la plataforma a través de grupos o clústers de nodos agrupados en una serie de servicios genéricos, de los que se indica la funcionalidad de cada uno, aunque la interfaz de acceso y la configuración son dependientes de la implementación empleada para ofrecer cada servicio.
- MaaS (Monitoring as a Service): En una plataforma descentralizada se necesita un mecanismo que mantenga actualizada la información sobre el estado global del sistema, la disponibilidad de los datos y los niveles de utilización de los recursos.

Siguiendo estos principios, el grupo ISTR ha desarrollado una plataforma (ver Ilustración 1) basada en un bus de datos compuesta por los siguientes servicios:

- Servicio de serialización: Resuelve la serialización de las instancias de los tópicos (unidad de información que se publica en el bus de datos) que se necesita para intercambiar información a través de una red o de un recurso de persistencia, con independencia del lenguaje de programación con el que se han codificado los agentes que acceden a él.
- Servicio de distribución: Resuelve el acceso seguro a una información global compartida en un sistema distribuido. Proporciona dos servicios claves: es el medio de

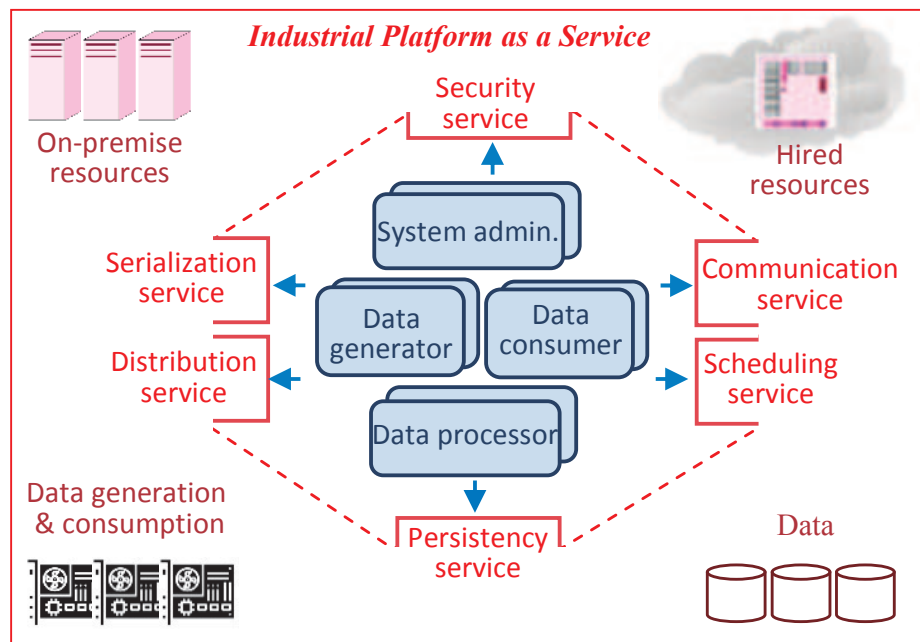


Ilustración 1: Plataforma como servicio basado en la arquitectura de referencia RAI4.0

registro y consulta de los metadatos que cualifican los tópicos y de los datos de configuración y coordinación de los servicios distribuidos.

- Servicio de comunicación: Facilita el registro de las instancias de los tópicos de la plataforma y la transferencia de instancias entre agentes del entorno en base al paradigma publicador/suscriptor. Este servicio gestiona el ciclo de vida de las instancias de los tópicos, escala la capacidad de gestión del volumen de datos en base a particiones y garantiza la distribución de los datos en base a la declaración de grupos de subscriptores. Todos estos aspectos se configuran en base a metadatos que se asocian a las descripciones de los tópicos.
- Servicio de planificación: Planifica las ejecuciones de las tareas de procesamiento en los nodos en que están almacenados los datos, bien por transferencia e invocación de los códigos de procesamiento, o por activación de los componentes software que están instanciados en los correspondientes nodos.
- Servicio de persistencia: Almacena las instancias de datos seleccionadas para que su ciclo de vida supere el nivel de persistencia establecido en el servicio de comunicación general para las instancias del tópico. En una plataforma big data hay que balancear las características de consistencia, disponibilidad y particionado de los datos por lo que hay que tener en consideración los diferentes paradigmas NoSQL.
- Servicio de seguridad: Garantiza la autenticación de los agentes y la integridad, confidencialidad y disponibilidad de la información. En una plataforma Big Data las reglas de seguridad deben ser dinámicas y estar orientadas al dato, esto es, los metadatos asociados a cada tópico deben incluir los criterios sobre quién y cómo puede acceder al dato y bajo qué restricciones.

Estos servicios permiten dar soporte a una serie de tópicos que pueden ser el desencadenante o el resultado de la ejecución de un workflow, ya que como se ha dicho, el dato es el centro de la arquitectura. Para este proyecto en concreto se utilizan los servicios

de distribución, comunicación y planificación, empleando en concreto Zookeeper, Kafka y Spark. Estas tecnologías, entre otras, se explican en el próximo apartado.

3.2. Tecnologías empleadas

Para llevar a cabo la implementación del proyecto es necesario hacer uso de diferentes tecnologías. En este apartado, se explican algunas de las empleadas junto con su función dentro del sistema. Para facilitar la comprensión de este apartado se muestra, en la Ilustración 2, de forma esquemática, cómo se organizan y comunican las diferentes tecnologías entre sí. El apartado se divide en tecnologías big data, lenguajes de programación y frameworks de desarrollo, librerías de machine learning e infraestructura.

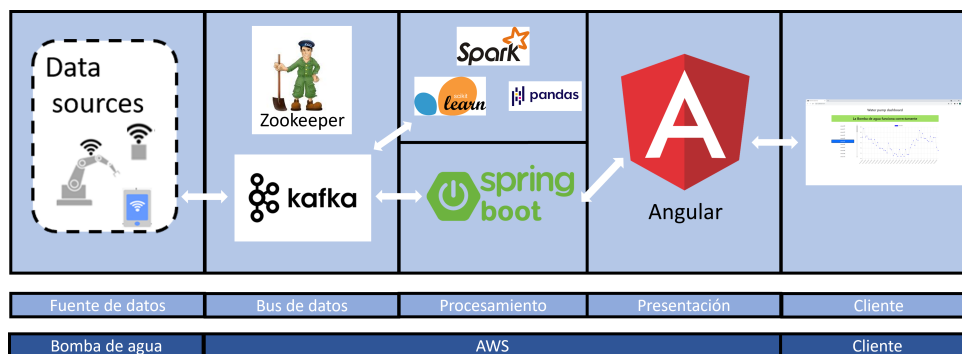


Ilustración 2: Organización de tecnologías en el sistema de mantenimiento predictivo

3.2.1. Lenguajes de programación y frameworks de desarrollo

Para desarrollar los diferentes elementos que forman parte de este proyecto, como pueden ser los modelos predictivos, los workflows de preprocesamiento o la interfaz gráfica empleada para monitorizar el estado de la máquina en tiempo real ha sido necesario emplear varios lenguajes de programación y frameworks de desarrollo. Las opciones escogidas fueron las siguientes:

JAVA [9]

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems. Funciona bajo un paradigma de programación orientado a objetos y corre sobre una máquina virtual, lo que hace que sea un lenguaje con el que se pueden desarrollar aplicaciones multiplataforma, es decir, que funcionen independientemente del hardware subyacente.

En este proyecto se ha utilizado para adaptar la implementación del software Stream Ingestor, que permite leer y publicar datos de un fichero csv en Kafka con un throughput indicado; y desarrollar un servicio con Spring boot que conecte el dashboard con el bus de datos Kafka para poder realizar la visualización en tiempo real de las medidas y las predicciones realizadas sobre ellas.

Python [10]

Python es un lenguaje de programación interpretado, multiparadigma y multiplataforma, gestionado por Python Software Foundation. Su filosofía se centra en la legibilidad

y simplicidad del código. Otra de sus características principales es la extensibilidad del código, pudiendo desarrollar y adaptar librerías software en C y C++, consiguiendo una enorme versatilidad y un buen rendimiento. En la actualidad es uno de los lenguajes más empleados dentro del ámbito de la ciencia de datos y la inteligencia artificial.

En este proyecto se utiliza Python para realizar el análisis inicial de los conjuntos de datos y generar los bloques de código y modelos predictivos desplegables en Spark.

Typescript [11]

Typescript es un lenguaje de programación que añade tipos estáticos y objetos a basados en clases a Javascript. Extiende la sintaxis de Javascript y está pensado para grandes proyectos y para ser compilado y traducido a código Javascript. Es un lenguaje de código abierto desarrollado y mantenido por Microsoft, que puede ser usado para aplicaciones que se ejecuten tanto en el cliente como en el servidor.

Spring Boot [12][13]

Es un framework de desarrollo de aplicaciones construido sobre la plataforma Java EE. Permite desarrollar aplicaciones de forma sencilla mediante anotaciones y se ha convertido en una alternativa al modelo EJB (Enterprise JavaBeans).

Soporta múltiples paradigmas de programación: basado en microservicios, programación reactiva, programación en la nube, aplicaciones web, serverless, programación dirigida por eventos o por lotes.

Angular [14]

Angular es un framework de desarrollo de aplicaciones web desarrollado en TypeScript. Se utiliza para crear aplicaciones web de tipo single-page application, bajo un paradigma de programación reactiva.

Las aplicaciones de tipo single-page application se caracterizan por tener una única página compuesta de varios elementos o componentes que son intercambiados por un módulo de routing en función de las interacciones realizadas por los usuarios. Cada componente tiene una parte de estructura desarrollada en HTML y una parte desarrollada en JavaScript con propiedades y funciones que se pueden enlazar a los elementos HTML.

La programación reactiva consiste en que el código se reevalúa cuando alguna variable cambia su valor, de manera que, por ejemplo, podemos indicar la siguiente expresión $a=b+c$ y, cada vez que b o c cambien de valor, se reevaluará la expresión y cambiará el valor de la variable a . Esto facilita el trabajo con flujos de datos y peticiones asíncronas.

3.2.2. Tecnologías big data

Como hemos dicho antes, ha sido necesario emplear diferentes tecnologías big data para desplegar la plataforma que pueda dar soporte al sistema de mantenimiento predictivo. A continuación, se realiza una breve descripción de las tecnologías empleadas y se explica cuál es el papel concreto de cada una de ellas.

Kafka [15]

Apache Kafka es un software libre de distribución de mensajes mediante un paradigma publicador-suscriptor. Este software es ampliamente usado para realizar tareas de comunicación entre sistemas, analíticas de flujos de datos y pipelines de datos de alto rendimiento. Se basa en la creación de tópicos, que representan cada flujo de datos y pueden estar manipulados por dos roles principalmente: publicador y suscriptor. El primero es el encargado de publicar o introducir los eventos en el tópico, y el segundo se encarga de recoger y leer cada uno de los eventos de los tópicos a los cuales está suscrito. Existen interfaces para diferentes lenguajes de programación como, por ejemplo, Java y Python. En este proyecto, Kafka se utiliza para implementar el bus de datos al que se van a conectar todas las aplicaciones para intercambiar información.

Zookeeper [16]

Apache Zookeeper es un servicio que gestiona el mantenimiento de la información de configuración, sincronización, nombres y grupos de servicios distribuidos. Este software sirve para dar soporte a las necesidades de coordinación y sincronización de otros servicios distribuidos, además permite gestionar autorización y autenticación de servicios, por lo que sirve para implementar un primer nivel de seguridad.

En este trabajo se utiliza Zookeeper para dar soporte a Kafka (Servicio de comunicación) y a Spark (Servicio de planificación), de manera que pueda funcionar de manera sincronizada.

Spark [17]

Apache Spark es un motor de analítica para realizar procesamiento de datos a gran escala. Permite realizar análisis de datos en batch (similar a MapReduce), pero además cuenta con una versión Spark Streaming para procesar flujos continuos de datos. Ofrece interfaces de programación en Java, Scala, Python, R, y SQL, para poder desarrollar aplicaciones paralelas mediante comandos de alto nivel.

En este proyecto se utiliza Spark como servicio de planificación para desplegar las tareas de procesamiento de datos y machine learning. Aquí es donde se va a centrar la mayor parte del procesamiento de los datos que van llegando a la plataforma. Se ha escogido Spark, ya que existen librerías de integración entre Kafka y Spark y soporta el desarrollo de aplicaciones en Python, lenguaje en el que se ha realizado el procesamiento de datos y la creación de modelos predictivos.

3.2.3. Librerías de machine learning & data science.

El preprocesamiento de datos y la generación de modelos predictivos se ha realizado utilizando librerías de software que incluyen las funciones más comúnmente utilizadas para trabajos de data science. A continuación, se mencionan las utilizadas:

Keras [18]

Keras es una API de deep learning construida sobre tensorflow 2.0 con el objetivo de hacer que el proceso de construcción y prueba de modelos sea más simple y rápido. Implementa una interfaz sencilla y consistente, que permite realizar el desarrollo con un menor número de acciones e intervenciones del usuario. Tiene disponibles un gran número

de algoritmos y métricas, pero permite también que el usuario implemente sus propias versiones de los algoritmos. Incluye modelos preentrenados y datasets de prueba como los clásicos MNIST o CIFAR10 y CIFAR100, que son de gran utilidad para introducirse en el manejo de Keras o probar nuevos algoritmos sobre datasets probados y fiables.

Scikit-learn [19]

Scikit learn es una librería open source de machine learning para Python construida sobre SciPy, NumPy y Matplotlib. Es una de las librerías más utilizadas en el ámbito del análisis de datos debido a su madurez, sencillez y eficiencia. Implementa funciones de clustering, clasificación, regresión, así como de reducción de la dimensionalidad, preprocesado y visualización de datos.

En este trabajo se ha utilizado para realizar el procesado de los datos previo a la construcción de modelos predictivos. Además, se ha utilizado también para construir los modelos de clasificación basados en árboles de decisión o vectores soporte.

Matplotlib [20]

Matplotlib es una librería de Python que permite visualizar conjuntos de datos por medio de diferentes representaciones de datos para conocer su distribución, analizar tendencias o estudiar correlaciones entre las diferentes variables, entre otras cosas. Permite, asimismo, realizar figuras compuestas por varios gráficos, personalizar los estilos y exportar los resultados a un fichero.

En este trabajo se ha utilizado matplotlib para realizar un análisis inicial de los conjuntos de datos, así como para generar algunas de las figuras que se muestran en este documento.

Pandas [21]

Pandas es una librería de Python especializada en manipular conjuntos de datos. Mediante sus objetos Series y DataFrame, permite realizar una gran cantidad de operaciones sobre grandes tablas de datos de forma realmente eficiente. Facilita al desarrollador las tareas de gestión de los conjuntos de datos y permite leer y escribir ficheros en diferentes formatos. Proporciona pipelines que permiten agrupar varias transformaciones para aplicarlas todas en un solo paso.

En este trabajo se ha utilizado Pandas para manipular los datos de los conjuntos de entrenamiento, realizar muestreos, leer los ficheros de datos originales y escribir ficheros con datos procesados.

3.2.4. Infraestructura

El sistema implementado se ha desplegado sobre dos infraestructuras físicas: un clúster de máquinas virtuales en un ordenador personal y un clúster de máquinas virtuales aprovisionadas en Amazon Web Services (AWS).

Ordenador personal

Durante la fase inicial, el entrenamiento de algunos modelos predictivos y el desarrollo del código se hizo en un ordenador personal. Asimismo, se realizó el despliegue de la

plataforma mediante el uso de maquinas virtuales con una distribución de Linux instalada para las pruebas. Este cuenta con las siguientes especificaciones:

- **Procesador:** AMD Ryzen 2600 6-core 3,4GHz
- **Memoria:** 16GB DDR4 2400 MHz
- **Disco:** 500GB SSD NVMe

Además, se ha utilizado como host Ansible para realizar las tareas de aprovisionamiento y configuración de recursos en AWS.

AWS [22]

Amazon Web Services (AWS) es un conjunto de servicios de computación en la nube ofrecidos por Amazon. Dentro del paradigma de computación en la nube se centra en Infrastructure as a Service (IaaS) y en Platform as a Service (PaaS). Entre sus servicios más destacados se encuentra Elastic Compute (EC2)[23], que permite realizar aprovisionamiento de máquinas virtuales de diferentes tipos, desde máquinas muy básicas y baratas hasta máquinas con hardware de última generación; Elastic Block Storage (EBS)[24] que permite realizar aprovisionamiento de unidades de almacenamiento, tanto de estado sólido como discos duros, que se pueden acoplar a las máquinas de EC2; y Simple Storage Service (S3)[25], que es un sistema de almacenamiento de ficheros en la nube.

En este proyecto se han usado EC2 y EBS para desplegar los diferentes servidores que componen la plataforma y poder realizar las pruebas de laboratorio. Para el despliegue de la plataforma se han empleado nodos con las siguientes características:

- **Procesador:** AMD EPYC 7000 @ 2,5GHz x 4
- **Memoria:** 16GB DDR4
- **Disco:** 8GB SSD

Ansible [26]

Ansible es un sistema de automatización que permite configurar equipos informáticos de manera remota. Se trata de un sistema sin cliente, es decir, la configuración se realiza desde un host mediante ssh y no requiere instalar nada en el resto de los equipos. Para que Ansible sea capaz de configurar los equipos es necesario definir una receta o playbook en formato yaml en la que se detallen el estado que debe tener el equipo al finalizar, indicando, por ejemplo, qué software debe instalarse.

En este proyecto se ha utilizado para realizar de forma automatizada la configuración de los equipos donde se van a desplegar los diferentes servidores de la plataforma. Debido a que parte del despliegue se realiza en la nube, Ansible permite reducir costes eliminando los recursos al finalizar las pruebas para que dejen de generar gasto y poder reconfigurarlos de forma rápida cuando se necesiten de nuevo.

4 Metodología de trabajo

4.1. Metodología de trabajo del proyecto y planificación general

En esta sección, se explica la planificación seguida para la realización del proyecto. Para ello se dispone de un diagrama Gantt (ver Ilustración 3) en el que se muestran las diferentes tareas del proyecto, la duración de cada uno de ellas, las dependencias entre las tareas y los hitos más importantes. A continuación se detalla cada una de las tareas del proyecto.

Establecimiento de objetivos

El fin de esta tarea era fijar la temática del trabajo y establecer los objetivos a cumplir. Para ello se dedicó la primera semana de trabajo.

Búsqueda de información, estudio y análisis

Esta segunda tarea del proyecto con una duración aproximada de 2 meses y medio tenía por objeto encontrar y estudiar todo lo relativo a la aplicación de minería de datos en entornos big data: técnicas, algoritmos, técnicas de visualización, procesamiento de datos, streaming, etc, así como, herramientas y tecnologías que sirvieran para implementar las técnicas aprendidas. Para ello realicé una serie de cursos online de python[10] y diferentes paquetes de software enfocados a la minería de datos y recopilé las diferentes técnicas en varios jupyter notebooks.[27]

Por último, dediqué unos días a buscar trabajos relacionados en la literatura, para tomar de guía y así poder proporcionar avance a partir de lo existente en el estado del arte.

Implementación del prototipo

Esta tarea comprende todo el desarrollo de software del proyecto y se divide en dos partes. La primera parte es la que más trabajo ha requerido y se corresponde con el desarrollo del proceso de minería de datos siguiendo la metodología CRISP-DM [28] que se explica en el siguiente apartado. El objetivo de esta primera parte era analizar el conjunto de datos disponible para construir y desplegar un sistema distribuido capaz de realizar predicciones en tiempo real; la segunda parte era realizar una sencilla interfaz web capaz de mostrar los valores registrados por los sensores y el estado predicho de la máquina en tiempo real.

Redacción del documento escrito

La tarea de redacción de la memoria se ha realizado en paralelo con el resto de las tareas. Esto es, mientras que se recogía información se redactaba algún apartado como el de trabajos relacionados o tecnologías que se iban a emplear. Lo mismo ocurrió en el proceso de implementación, al tiempo que se iban consiguiendo nuevos avances se documentaban aprovechando así los momentos en los que los modelos predictivos estaban en proceso de entrenamiento.

En esta tarea existen dos hitos importantes: el primero, la entrega del primer borrador del documento a la directora del trabajo para revisar algunos apartados ya redactados y establecer la estructura del final del documento, el segundo el depósito del manuscrito en secretaría para solicitar la formación de un tribunal que lo evalúe.

Preparación de la defensa

Esta es la última fase que comienza tras depositar el trabajo en secretaría. El objetivo es preparar la presentación con la que defender el trabajo ante un tribunal para su calificación.

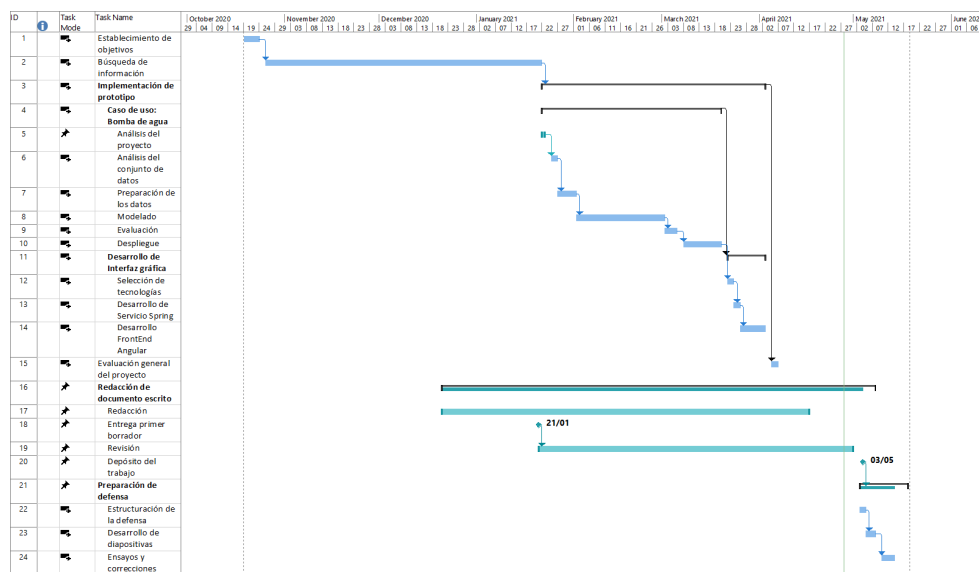


Ilustración 3: Diagrama gantt del proyecto de minería de flujos de datos

4.2. Metodologías para data minería de flujos de datos

Al ser relativamente nuevo para mi el mundo de la minería de datos, el primer paso para poder afrontar un proyecto de estas características fue realizar un estudio de las diferentes metodologías de trabajo, de manera que pudiera desarrollar mi proyecto siguiendo las buenas prácticas de la disciplina. En la literatura encontré diversas metodologías de trabajo que describo sucintamente a continuación.

4.2.1. KDD

KDD es el acrónimo de Knowledge Discovery in Databases. Esta fue la primera metodología de trabajo estructurada en minería de datos. Se refiere al proceso de identificar patrones válidos, novedosos, potencialmente útiles y principalmente entendibles.[29]. En [29], Usama et al. muestran KDD como un proceso dividido en 5 fases: Selección, Preprocesado, Transformación, Minería de datos y Evaluación/Interpretación; en el que la entrada son datos y la salida conocimiento.

Ventajas

- Fue la primera iniciativa de estandarizar un proceso para la extracción de conocimiento.
- Establece varias fases diferenciadas para conseguir información relevante a partir de los datos.

Inconvenientes

- No tiene en cuenta el estudio del contexto y los objetivos del negocio.
- No cuenta con una fase de puesta en producción. No establece ningún protocolo de actuación una vez terminada la fase de Evaluación/Interpretación.

4.2.2. SEMMA

SEMMA[30] es una metodología de minería de datos definida por la empresa SAS. En esta metodología el proceso de minería de datos se divide en 5 fases: Muestreo, Exploración, Modificación, Modelado y Evaluación (SEMMA por sus siglas en inglés). A continuación, se explica brevemente las etapas de SEMMA:

- Muestreo: es la primera fase y consiste en conseguir una muestra representativa del conjunto de datos con la que trabajar. El objetivo es obtener un conjunto de datos más pequeño para poder procesarlo, pero que sea tan fiel al original como sea posible.
- Exploración: sigue al muestreo y consiste en estudiar los conjuntos de datos resultantes para detectar distribuciones probabilísticas, relaciones o problemas en los datos.
- Modificación: en esta fase se crean y seleccionan las características más representativas y relevantes de conjunto de datos y se eliminan las que no aportan información.
- Modelado: en esta fase se emplean los conjuntos de datos y herramientas analíticas para generar modelos de predicción.
- Evaluación: En esta fase se analizan los datos obtenidos para evaluar la usabilidad y fiabilidad de los modelos obtenidos del proceso de minería de datos.

Ventajas

- Impulsada por una empresa con experiencia en el análisis de datos.
- Añade la fase de muestreo lo que permite abordar conjuntos de datos de mayor magnitud.
- Añade una fase de exploración de datos y agrupa las de transformación de datos en una única fase.

Inconvenientes

- Igual que KDD, no tiene en cuenta el entorno y los intereses del proyecto y se centra únicamente en el proceso de análisis de los datos.
- Las fase de despliegue y documentación no están incluidas dentro del proceso.

4.2.3. CRISP-DM

CRISP-DM es una metodología de análisis de datos desarrollada por el CRISP-DM consortium, un consorcio de grandes empresas con experiencia en minería de datos, y es uno de los modelos de proceso más empleados actualmente. CRISP-DM es un acrónimo de *CRoss Industry Standard Process for Data Mining* [31]. Consta de 6 fases que se explican a continuación:

- Entendimiento del proyecto: en esta fase se busca encontrar el beneficio, así como los riesgos potenciales de realizar el proyecto; se toma una decisión sobre si el proyecto se considera apropiado, necesario y será la que marque el rumbo del resto del proyecto, ya que se deben dejar claros los objetivos y las soluciones esperadas.
- Entendimiento de los datos: en esta fase se debe comprobar que tenemos suficientes datos y que son relevantes para el proyecto, aunque no se puede saber con precisión hasta el final. Este es el momento en que se debe realizar un análisis visual de los datos, buscar outliers, correlaciones y, en definitiva, juzgar si la calidad y cantidad de los datos es suficiente para continuar con el proyecto y cumplir las expectativas iniciales. En caso de que no sea posible o solo se pueda realizar parcialmente, se deberá cancelar el proyecto o redefinir el alcance establecido en la fase anterior.
- Preparación de los datos: en esta fase se realizan todas las modificaciones necesarias en el conjunto de datos para prepararlo antes de la fase de modelado. Esta fase incluye corregir problemas detectados en la fase de entendimiento de los datos como inconsistencias en variables categóricas, eliminar outliers, tratamiento de valores nulos, normalización de los datos. Además, se realiza el proceso de creación y selección de características, proceso que tiene un doble objetivo: eliminar el ruido y utilizar únicamente los datos que aportan valor, y reducir los recursos necesarios y el tiempo de procesamiento del modelo.
- Modelado: en esta fase se escoge las herramientas y algoritmos de modelado que más se adapten al problema y nos ayuden a obtener conocimiento. Hay muchas opciones, desde herramientas que producen resultados fácilmente interpretables y permiten extraer reglas, hasta herramientas que funcionan como una caja negra, pero que en ocasiones tienen un rendimiento superior. Tras realizar el entrenamiento se debe

evaluar si los resultados son mejorables y en caso de que sea posible obtener mejores resultados se vuelve a la fase de preparación de datos para intentar conseguir un mejor desempeño. Por el contrario, si se cree que no es posible conseguir una mejora o que la mejora no va a ser sustancial, se pasa a la fase siguiente.

- **Evaluación:** en esta fase se evalúan los resultados, pero no desde el punto de vista técnico pues no se debería salir de la fase anterior hasta conseguir el mejor resultado posible. Lo que se busca en esta fase es evaluar los resultados desde el punto de vista del negocio o del proyecto para comprobar si se han cumplido los objetivos establecidos en la primera fase. Al igual que en la fase de entendimiento de los datos, si no se ha cumplido se debe cerrar el proyecto, si se cumple una parte se deben revisar los objetivos y si se consigue, se pasa a la última fase.
- **Despliegue:** en esta fase se deben desarrollar y desplegar todos los componentes necesarios para poner la solución creada en producción. Otras acciones que suelen realizarse con frecuencia en esta fase son la realización de informes, o el despliegue de un sistema de monitorización para revisar cómo evoluciona el rendimiento del modelo, entre otras.

Ventajas

- Desarrollado por un consorcio de empresas con experiencia en el análisis de datos.
- El proceso de análisis comienza con un proceso de entendimiento del problema y el negocio, que sirve como punto de partida para el resto del análisis.
- Se incluye el proceso de despliegue de los modelos en conjunto con el resto del sistema como parte final del proyecto.

Inconvenientes

- No está pensado para flujos de datos donde es posible que se necesite realizar un despliegue completo de la arquitectura durante la fase de modelado para comprobar la evolución del modelo con la llegada de nuevos datos, tanto a nivel de consumo de recursos y capacidad, como a nivel de rendimiento, ya que el entrenamiento continua con cada nueva instancia de datos.

Como se puede comprobar esta es una alternativa más completa que las anteriores, ya que incluye una fase de comprensión del proyecto previa al análisis de los datos. Esto permite centrar el foco del trabajo en los objetivos reales del negocio y detectar de manera temprana si los datos son adecuados para cumplir estos objetivos. Por este motivo y porque es la alternativa más empleada en la actualidad se escogió esta metodología para la realización de los experimentos.

5 Caso de estudio

En este capítulo se describe un caso de estudio en el que se aplica todo el proceso CRISP-DM a un conjunto de datos para construir un predictor que identifique cuándo las bombas de agua que abastecen un pueblo pueden fallar. El apartado se estructurará en diversos apartados, uno por cada fase de CRISP-DM.

5.1. FASE 1: Entendimiento del problema

El conjunto de datos objeto de estudio corresponde a los datos recogidos por una serie de sensores instalados en una bomba de agua responsable de abastecer a un pueblo pequeño[32]. La bomba de agua está situada fuera del pueblo, por lo que un fallo puede suponer un corte de suministro prolongado ya que requiere un desplazamiento para diagnosticar la avería y repararla. Por tanto, el propietario quiere aplicar una solución basada en minería de datos para detectar fallos lo antes posible y poder reducir el tiempo en el que la bomba de agua esté fuera de servicio. Para ello, se tratará de construir un sistema que analice el flujo de datos generado por los sensores instalados en la máquina con el objeto de detectar los fallos que se produzcan y generar alarmas que permitan atajarlos con la mayor rapidez posible. Esto es, desarrollar un modelo predictivo capaz de procesar datos en tiempo real y determinar si la máquina está funcionando con normalidad o si, por el contrario, presenta algún malfuncionamiento o bajada de rendimiento que haga prever un fallo de la bomba. Al encontrarse la bomba alejada del pueblo, ir a arreglarla supone realizar un desplazamiento y, por tanto, tiene un coste tanto en tiempo como en dinero. Un hecho importante es que los falsos positivos deben minimizarse porque se produciría una parada de la bomba de agua desde que se detecta la anomalía hasta que se comprueba y repara para evitar posibles agravamientos de la avería, a pesar de no existir ninguna avería real. Esto hace que las métricas más importantes a la hora de reducir el tiempo sin servicio y el coste sean la accuracy y la tasa de falsa alarma.

En la descripción del conjunto de datos, lamentablemente, no se indica a qué variable corresponden las medidas de cada uno de los diferentes sensores, ni las unidades en las que se expresa cada magnitud. Por lo tanto, no se pueden extraer reglas comprensibles de ellos y solo se puede hacer un tratamiento meramente numérico.

5.2. FASE 2: Entendimiento del conjunto de datos

Los datos aportados por el propietario de la bomba de agua se encuentran en formato csv. El fichero contiene las lecturas realizadas cada segundo durante 4 meses de funcionamiento. Cada lectura contiene la fecha y hora de la medición, el estado de la bomba en ese instante y las lecturas de 52 sensores.

El primer paso consiste en analizar los datos y realizar unas estadísticas que permitan

conocer el rango de valores que comprende cada una de las variables. En las ilustraciones 4 y 5 se muestran, como ejemplo, una vista previa de las 5 primeras filas de datos y una descripción estadística básica de algunas variables en la que se computa la media, moda, percentiles, mínimos y máximos del conjunto de datos. La información completa se encuentra disponible en los jupyter notebooks del repositorio de github [27].

Unnamed: 0	timestamp	sensor_00	sensor_01	sensor_02	sensor_03	sensor_04	sensor_05	sensor_06	sensor_07	sensor_08	sensor_09	sensor_10	
0	0	2018-04-01 00:00:00	2.465394	47.09201	53.2118	46.310760	634.3750	76.45975	13.41146	16.13136	15.56713	15.05353	37.22740
1	1	2018-04-01 00:01:00	2.465394	47.09201	53.2118	46.310760	634.3750	76.45975	13.41146	16.13136	15.56713	15.05353	37.22740
2	2	2018-04-01 00:02:00	2.444734	47.35243	53.2118	46.397570	638.8889	73.54598	13.32465	16.03733	15.61777	15.01013	37.86777
3	3	2018-04-01 00:03:00	2.460474	47.09201	53.1684	46.397568	628.1250	76.98898	13.31742	16.24711	15.69734	15.08247	38.57977
4	4	2018-04-01 00:04:00	2.445718	47.13541	53.2118	46.397568	636.4583	76.58897	13.35359	16.21094	15.69734	15.08247	39.48939

Ilustración 4: Vista de las 5 primeras filas del conjunto de datos

Unnamed: 0	sensor_00	sensor_01	sensor_02	sensor_03	sensor_04	sensor_05	sensor_06	sensor_07	sensor_08
count	220320.000000	210112.000000	219951.000000	220301.000000	220301.000000	220301.000000	215522.000000	214869.000000	215213.000000
mean	110159.500000	2.372221	47.591611	50.867392	43.752481	590.673936	73.396414	13.501537	15.843152
std	63601.049991	0.412227	3.296666	3.666820	2.418887	144.023912	17.298247	2.163736	2.201155
min	0.000000	0.000000	0.000000	33.159720	31.640620	2.798032	0.000000	0.014468	0.000000
25%	55079.750000	2.438831	46.310760	50.390620	42.838539	626.620400	69.976260	13.346350	15.907120
50%	110159.500000	2.456539	48.133678	51.649300	44.227428	632.638916	75.576790	13.642940	16.167530
75%	165239.250000	2.499826	49.479160	52.777770	45.312500	637.615723	80.912150	14.539930	16.427950
max	220319.000000	2.549016	56.727430	56.032990	48.220490	800.000000	99.999880	22.251160	23.596640

Ilustración 5: Descripción estadística de las variables del conjunto de datos

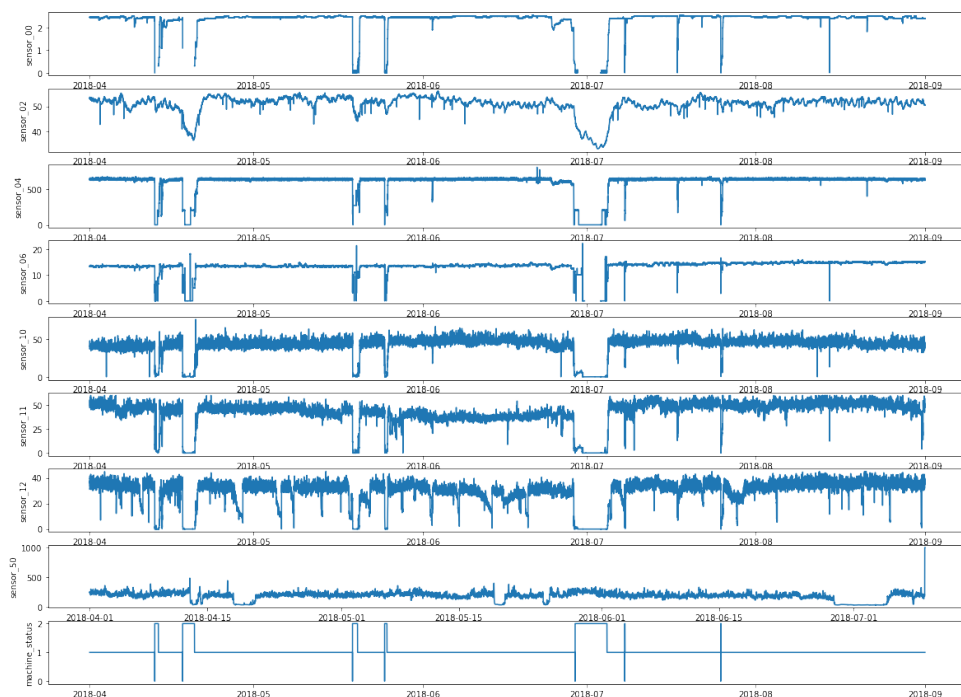


Ilustración 6: Evolución temporal de los valores enviados por diferentes sensores y el estado de la máquina

El conjunto de datos cuenta con 53 variables, por lo que, con objeto de ver si es factible reducir el número de variables de entrada se procede a realizar un análisis de

correlación de las variables respecto a la variable objetivo. Del resultado se observa que algunas variables que están fuertemente correlacionadas con la variable objetivo, mientras que otras son prácticamente independientes. Además, generamos un mapa de calor (ver Ilustración 7) de correlación entre cada par de variables. Esto nos permite ver que muchas de las variables muestran información redundante y, por tanto, podemos aplicar técnicas de reducción de la dimensionalidad, de manera que se reduzca el tiempo de entrenamiento de los modelos predictivos sin tener una pérdida de información significativa, además de construir modelos más sencillos de interpretar.

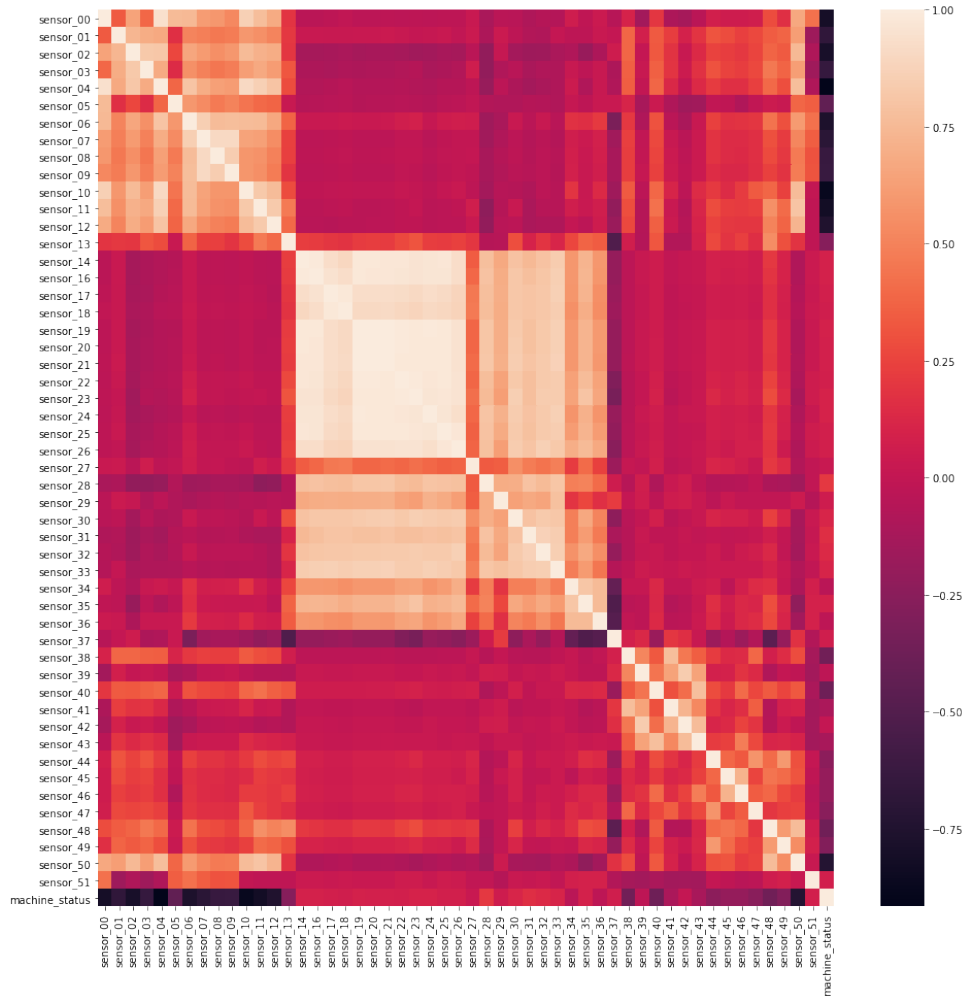


Ilustración 7: Correlación entre variables representada mediante mapa de calor

Al analizar los datos se observa, también, que no existen datos del sensor 15, por lo que debe haber algún tipo de problema con el sensor y se debe descartar esta variable. Por último, se analiza la variable objetivo.

La variable objetivo es la columna `machine_status`. Se trata de una variable categórica que cuenta con 3 valores posibles: `NORMAL`, `BROKEN` y `RECOVERING`. En la ilustración 8 se muestra la cantidad instancias de disponibles de cada clase. Como se puede ver, la muestra está poco equilibrada y el número de registros en los que el estado es `NORMAL` (93,4 %) es muy alto en comparación con el resto de los estados. Esto es lógico, ya que la mayor parte del tiempo la máquina debe presentar un funcionamiento normal, pero hace que contemos con pocas muestras del caso `BROKEN`(0,003 %) y `RECOVERING`(6,5 %) para la fase de entrenamiento y puede hacer que el modelo quede sesgado. Este hecho conduce a elegir la tasa de falsa alarma como una medida de evaluación del modelo necesaria,

ya que la accuracy puede llevar a engaño. Por ejemplo, un modelo que siempre prediga un estado NORMAL tendrá más del 90 % de accuracy, sin embargo, será totalmente inútil y jamás detectará un fallo.

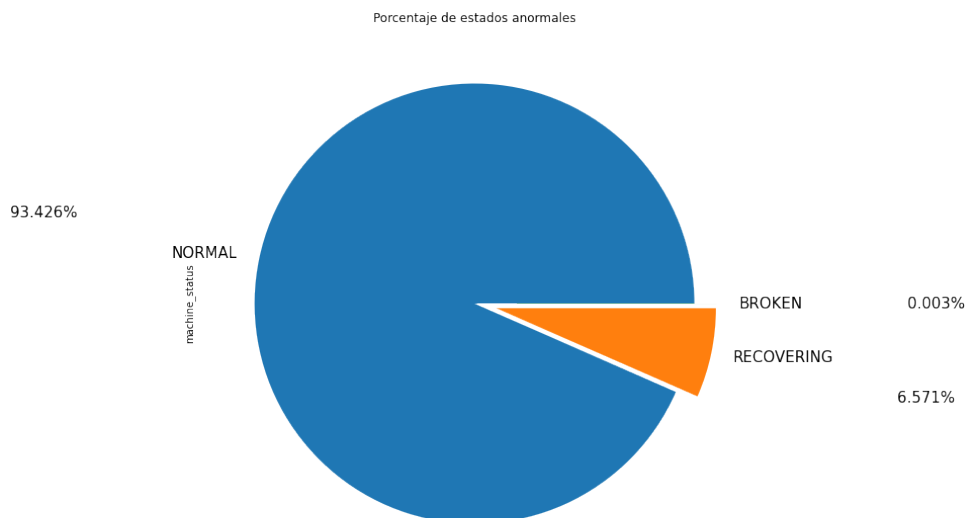


Ilustración 8: Porcentaje de instancias del conjunto de datos que hay para cada estado de la bomba

5.3. FASE 3: Preparación de los datos

En esta fase se preparan los datos para poder encarar la fase de construcción de modelos. En primer lugar, se eliminan las columnas que no aportan información: Unnamed: 0 (autogenerada por pandas al cargar el csv) y sensor_15 (solo contiene valores nulos).

El siguiente paso fue transformar la variable objetivo en una variable binaria. Dado que se cuenta con muy pocos registros del estado BROKEN, se consideran los estados RECOVERING como estados anormales y consecuencia de un estado BROKEN previo. Por lo que todos los valores RECOVERING se transforman en BROKEN. De esta forma, el problema se convierte en un problema de clasificación binaria, en el que hay una mayor presencia del estado BROKEN, que es el que realmente nos interesa detectar.

Se procede a continuación a particionar el conjunto de datos para proceder al entrenamiento y la prueba de los predictores. Para ello se divide la muestra en dos conjuntos de datos, uno para el entrenamiento y el otro para la validación de los modelos. Hay varias estrategias como el muestreo aleatorio o el muestreo estratificado. Sin embargo, al tratarse de una serie temporal y los datos se generan de manera consecutiva, se parte la muestra en dos trozos, ya que no tiene sentido incluir instancias más antiguas en el conjunto de validación que en el de entrenamiento. Para ello, ordenamos los registros de forma ascendente por fecha de lectura y seleccionamos un 60 % para entrenamiento y un 40 % para validación. Tras hacerlo comprobamos que ambos conjuntos contienen registros correspondientes a las diferentes clases de la variable objetivo.

Otro aspecto importante es decidir qué hacer con los valores nulos, ya que estos no se pueden procesar y provocarán errores al momento de entrenar y realizar predicciones. Para el tratamiento de valores nulos se estudiaron diferentes alternativas: eliminar los valores nulos, imputar el valor medio, moda, valor anterior o regresión lineal. Eliminar los registros con valores nulos hacía que se perdiera mucha información como se puede

ver en la ilustración 9. La regresión lineal supone utilizar un modelo predictivo diferente para cada variable y, por tanto, añadir un overhead al preprocesamiento de los datos. En un contexto de flujos continuos de datos, en el que se pretende contestar a las peticiones con la mayor rapidez posible esto no parece la mejor idea. Por tanto, nos queda la opción de imputar los valores como mejor alternativa, al tratarse de variables con un dominio continuo, la moda no tiene mucho sentido; y al tratarse de flujo de datos, el valor anterior solo sería aplicable en el caso del conjunto de entrenamiento, pues en producción no contaremos con los valores anteriores. Esto nos deja con la opción de valor medio como alternativa a utilizar.

Porcentaje de filas que contienen valores nulos

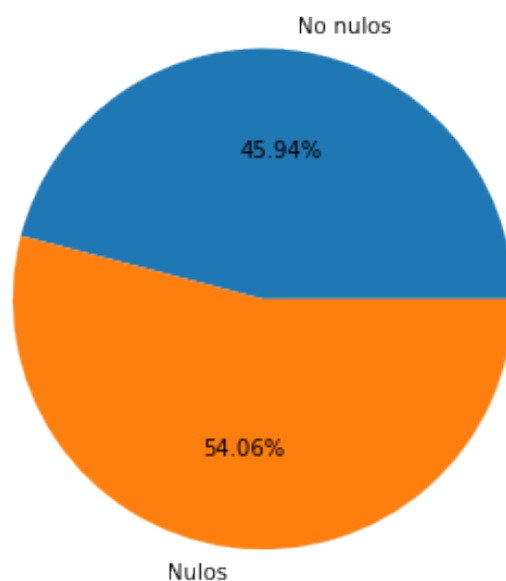


Ilustración 9: Proporción de filas que contienen algún valor nulo

A continuación se procede a seleccionar las variables a utilizar. Esto es necesario, ya que, si el número de variables es muy alto, el entrenamiento puede ser muy costoso en términos de memoria y tiempo. Además, un exceso de variables puede introducir ruido en el modelo y empeorar su capacidad de generalización. Como se ha visto en el paso de entendimiento de los datos hay algunas variables que guardan una fuerte correlación con la variable objetivo mientras otras parece que son independientes, estas últimas podrían resultar perjudiciales para el modelo.

Para la selección de variables se emplearon 4 alternativas: todas las variables, variables con coeficiente de correlación mayor que 0.7, componentes principales y k-best de sklearn. Con cada una de las alternativas se entrenarán los modelos y se escogerán las mejores combinaciones.

Todas las variables

Para este caso construiremos un pipeline de preprocesado de datos en el que solamente se eliminan las columnas que no aportan información `Unnamed: 0` y `sensor_15` y se aplicará el algoritmo para rellenar los valores nulos. Esto quiere decir que se generará un pipeline (Ilustración 10.a) que tomará los valores producidos por los sensores y los transformará en 50 parámetros que irán directamente a la entrada del modelo.

Variables más correlacionadas con el objetivo

En este caso se mantienen los pasos anteriores, pero, además se realiza un análisis de correlación previo para encontrar qué variables muestran un coeficiente de correlación superior a 0.7 y, se añade un nuevo paso al pipeline (Ilustración 10.b) encargado de eliminar el resto de las columnas.

Análisis de componentes principales

En este caso el objetivo es utilizar un análisis de componentes principales para encontrar combinaciones lineales de las diferentes variables, que permitan expresar la mayor parte de la variabilidad de conjunto de datos. En muchas ocasiones no es interesante aplicar este algoritmo ya que se pierde parte de la interpretabilidad al proyectarse las variables sobre un nuevo espacio, sin embargo, en este caso no sabemos a qué corresponde cada una de las variables por lo que no supone un problema añadido. Como este algoritmo tiene en cuenta la variación dentro de las variables puede verse afectada por la magnitud o las unidades de medida. Por tanto, añadirá una etapa de estandarización previa, quedando el pipeline que se muestra en la ilustración 10.c.

K-best

En este caso todo el pipeline es igual que en el caso anterior, pero se sustituye el algoritmo de análisis de componentes principales por el algoritmo K-best features del paquete Sklearn usando como medida chi-cuadrado. Por tanto, el pipeline quedaría como se muestra en la Ilustración 10.d.



Ilustración 10: Pipelines de preprocesamiento empleadas para la construcción de los modelos predictivos

5.4. FASE 4: Modelado

En esta fase se construyen diferentes modelos predictivos capaces de detectar los fallos en la bomba de agua. En este caso se han propuesto 4 alternativas de modelo predictivo: random forest, máquina de vector soporte, k-nearest neighbors y perceptrón multicapa.

5.5. FASE 5: Evaluación

Cada uno de ellos se ha combinado con los diferentes pipelines de preprocesamiento propuestos en la fase anterior y se han sometido a un ajuste de hiperparámetros mediante el método grid search. Además, para validar cada uno de los modelos generados durante la fase entrenamiento se ha utilizado el método 10-fold validation, ya que como dicen Kohavi et al. [33], 10-fold es un número suficiente para conseguir una estimación precisa de la capacidad de generalización del modelo. Con esto obtenemos los 16 casos que se muestran en la tabla 2, en la que se indican los valores de accuracy, TPR y FAR.

Además, en un intento de mejorar la capacidad de generalización del modelo, se ha generado un estimador de tipo ensemble con los 4 mejores modelos de la tabla anterior. Este estimador toma la predicción de cada uno de los modelos que lo componen y mediante votación selecciona la respuesta final. Los resultados obtenidos al evaluar los modelos se muestran en la tabla 2.

	Nombre	TPR	FAR	Accuracy
0	rfc_full	0.991967	0.002089	0.997617
1	rfc_corr	0.988754	0.000394	0.999070
2	rfc_pca	0.991967	0.002089	0.997617
3	rfc_kbest	0.991967	0.001289	0.998377
4	svc_full	0.990360	0.108606	0.896287
5	svc_corr	0.982786	0.003939	0.995404
6	svc_pca	0.973835	0.003784	0.995109
7	svc_kbest	0.584118	0.000024	0.979416
8	knc_full	0.986918	0.008332	0.991433
9	knc_corr	0.989213	0.000836	0.998672
10	knc_pca	0.965114	0.003307	0.995132
11	knc_kbest	0.990590	0.000859	0.998718
12	ann_full	0.993803	0.010362	0.989844
13	ann_corr	0.985311	0.003342	0.996097
14	ann_pca	0.973376	0.003569	0.995291
15	ann_kbest	0.997705	0.002602	0.997413
16	voting	0.99059	0.000716	0.998854

Tabla 2: Evaluación de los predictores construidos para los diferentes conjuntos de datos y algoritmos seleccionados

En esta fase seleccionamos los mejores modelos (señalados en negrita en la Tabla 2) entre los generados en el paso anterior y realizamos una última prueba para comprobar que cumplen con las necesidades planteadas en la primera fase. Ya tenemos unos modelos

capaces de detectar fallos en la bomba de agua con una fiabilidad bastante elevada, sin embargo, es importante tener en cuenta que se quiere desplegar en un sistema que se va a someter a un flujo continuo de datos. Por tanto, hay que medir el tiempo de respuesta para ver si el modelo es capaz de generar predicciones a la velocidad de generación de los datos. Para ello se ha añadido el tiempo que tarda en cada uno de los modelos en realizar diferentes predicciones y se ha calculado el tiempo promedio y la desviación estándar. Los resultados se muestran en la tabla 3.

	rfc_pca	rfc_kbest	knc_kbest	voting
count	88128.000000	88128.000000	88128.000000	88128.000000
mean	91.548125	29.042517	7.016570	110.944829
std	13.868766	4.130912	0.790092	6.258037
min	74.717600	26.133200	5.790100	101.457600
25 %	81.041700	27.911000	6.693100	107.484250
50 %	84.368550	28.505000	6.965450	109.407750
75 %	104.996650	29.536700	7.260200	112.353600
max	294.288700	187.521600	119.255400	283.676600

Tabla 3: Tiempo(ms) requerido por cada modelo para realizar una predicción

Los modelos muestran unas diferencias claras en los tiempos medios de predicción, siendo los dos que utilizan el algoritmo de preprocesado k-best los más rápidos. El que peor tiempo de respuesta tiene, como es lógico, es el modelo basado en votación ya que tiene que evaluar los parámetros utilizando diferentes estimadores y realizar la votación. De los dos modelos que emplean k-best, el más rápido es el basado en el algoritmo k-vecinos más cercanos, que es capaz de realizar las predicciones en un tiempo medio de 7,01 ms con una desviación típica de 0,7 ms y un valor máximo no muy lejano al valor medio del random forest con PCA. Por ello, parece que puede ser el más adecuado para un sistema en producción, pues cuenta con un 99,87 % de accuracy, un 0,008 % de FAR y un 99,05 % de TPR. Además, debido a la sencillez del algoritmo, es el que menos tiempo de entrenamiento ha requerido de todos los analizados.

5.6. FASE 6: Despliegue

En esta fase ya contamos con un modelo seleccionado de entre todos los candidatos evaluados y hay que desplegarlo con el resto de los componentes de la plataforma para comprobar que es capaz de atender las peticiones según lo esperado. Para ello se desplegará una plataforma como la que se ha mostrado en la ilustración 2 en el capítulo 3.

Este es el paso más complejo de todos ya que hay que configurar varios sistemas distribuidos para que trabajen de forma coordinada. A continuación, se explica el despliegue de los diferentes sistemas.

5.6.1. Infraestructura

Para realizar el despliegue de la infraestructura se han utilizado los servicios de computación bajo demanda EC2 de Amazon Web Services. El despliegue y configuración de las

instancias se ha realizado con el software Ansible. Para ello ha sido necesario seguir estos pasos:

Configuración en AWS

Para poder conectar Ansible a AWS y realizar el aprovisionamiento de los recursos necesarios, se necesita crear un usuario en el servicio de autenticación IAM. Con este usuario se pueden conseguir unas credenciales de acceso que hay que escribir en la configuración de Ansible. Tras esto, es necesario crear un par de claves para poder conectar con las instancias aprovisionadas sin que se solicite la contraseña del usuario. Por último, hay que crear un grupo de seguridad que permita la conexión a los puertos en los que escucharán los diferentes servicios.

Configuración del host de Ansible

En el nodo donde se vaya a ejecutar el cliente de Ansible hay que instalar Python y Ansible. Una vez hecho esto hay que instalar boto3 o botocore, una librería que permite conectarse con AWS como cliente para reservar recursos de forma dinámica y que es usada por Ansible. Por último, hay que copiar la clave privada para realizar la conexión ssh y las credenciales de acceso de AWS con los que se identificará Ansible.

Creación de playbooks de Ansible

Con todo el entorno listo para funcionar solo queda definir los playbooks que se van a ejecutar, el inventario de hosts a gestionar con Ansible y los ficheros con las variables necesarias.

En primer lugar, hay que crear el inventario. En este caso, como se va a realizar aprovisionamiento de máquinas y no se conoce la dirección IP de antemano, es necesario utilizar un plugin que permita gestionar inventarios dinámicos. Para ello hay que escribir en un fichero lo que se muestra en el Script 5.1

```
1 plugin: aws_ec2
2 regions:
3   - eu-west-1
4 filters:
5   tag:managed_by: ansible
```

Script 5.1: Fichero de configuración del inventario dinámico

A continuación, se muestran 3 playbooks: uno para el aprovisionamiento de la infraestructura (Script 5.2); otro para la configuración del entorno una vez se han desplegado los nodos (Script 5.3) y el tercero para liberar todos los recursos (Script 5.4).

```
1 - hosts: localhost
2   name: Setup Infrastructure and Provision Environment
3   remote_user: ec2-user
4   gather_facts: False
5
6   tasks:
7     - include_vars: config.yaml
8
9     - name: Create ec2 instances
```

```

10     ec2:
11         key_name: "{{ aws_ssh_key }}"
12         group: "{{ security_group }}"
13         instance_type: "{{ instance_type }}"
14         image: "{{ ami_id }}"
15         wait: true
16         exact_count: "{{ instance_count }}"
17         region: "{{ region }}"
18         zone: "{{ zone }}"
19         count_tag:
20             managed_by: ansible
21             name: "{{ instance_tag }}"
22         instance_tags:
23             managed_by: ansible
24             name: "{{ instance_tag }}"
25     register: ec2
26
27 - name: Get master node private ip
28   set_fact:
29       master_node: "{{ ec2.instances.0.private_ip }}"
30
31 - name: Write SSH config file
32   shell: |
33       echo "Host worker{{ item_idx }}" >> ssh_config
34       echo "    User ec2-user" >> ssh_config
35       echo "    Hostname {{ item.public_ip }}" >>
36           ssh_config
37       echo >> ssh_config
38   loop: "{{ ec2.instances }}"
39   loop_control:
40       index_var: item_idx
41
42 - name: Add all instance public IPs to host group
43   add_host: hostname={{ item.public_dns_name }} groups=
44       aws_ec2
45   loop: "{{ ec2.instances }}"
46
47 - name: Wait for SSH to come up
48   delegate_to: "{{ item.public_dns_name }}"
49   wait_for_connection:
50       delay: 60
51       timeout: 320
52   loop: "{{ ec2.instances }}"
53
54 - import_playbook: environment-play.yml

```

Script 5.2: Script Ansible para aprovisionar las máquinas virtuales de Amazon Web Services

En el Script 5.2 se observan 5 etapas diferenciadas. En la primera (líneas 9-25), se manda una petición a AWS para crear las instancias con las características indicadas en

el fichero config.yaml. En la segunda, se escribe el fichero de configuración para poder establecer conexiones mediante ssh posteriormente. En la tercera, se añaden los nodos desplegados al inventario de Ansible para poder ejecutar el resto de playbooks. En la cuarta, se realiza una espera para asegurarse de que el servicio ssh de los nodos aprovisionados está disponible antes de continuar. Por último, en la quinta etapa, se importa el playbook de configuración del entorno para que se ejecute en cuanto el servicio ssh esté disponible.

```
1 - name: Setup environment for experiment
2   hosts: aws_ec2
3   remote_user: ec2-user
4
5   tasks:
6     - include_vars: config.yaml
7     - name: Install the latest version of python
8       yum:
9         name: python3
10        state: latest
11        become: true
12
13    - name: Install the latest version of java
14      yum:
15        name: java
16        state: latest
17        become: true
18
19    - name: Install virtualenv
20      pip:
21        executable: "pip{{python.version}}"
22        name: virtualenv
23        become: true
24
25    - name: Copy requirements.txt
26      copy:
27        src: requirements.txt
28        dest: /home/ec2-user/requirements.txt
29
30    - name: Copy models directory
31      copy:
32        src: ../models
33        dest: /home/ec2-user
34
35    - name: Copy scripts directory
36      copy:
37        src: ../scripts
38        dest: /home/ec2-user
39
40    - name: Install experiment python dependencies
41      pip:
42        requirements: /home/ec2-user/requirements.txt
43        executable: "pip{{python.version}}"
```

```

44     become: true
45
46     - name: Extract apache platform into /home/ec2-user/
47       ansible.builtin.unarchive:
48         src: ~/Escritorio/repositorio/apache.tar.gz
49         dest: /home/ec2-user/
50
51     - name: Extract spark into /home/ec2-user
52       ansible.builtin.unarchive:
53         src: https://ftp.cixug.es/apache/spark/spark-3.1.1/
54           spark-3.1.1-bin-hadoop2.7.tgz
55         dest: /home/ec2-user/
56         remote_src: yes

```

Script 5.3: Script Ansible para preparar el entorno de trabajo

En el Script 5.3 se observan varias etapas en las que principalmente se instalan dependencias de software y se copian ficheros y directorios necesarios para desplegar la plataforma y los workflows. En primer lugar, se instalan Python y Java usando yum, el gestor de paquetes disponible en la distribución de Linux empleada en los nodos remotos. Tras esto, se copia un fichero de texto plano que contiene todas las librerías de Python empleadas en el workflow, y los directorios con los modelos predictivos y los scripts de los workflows a desplegar en Spark. A continuación, se instalan las librerías del fichero requirements.txt usando la versión de pip correspondiente a la versión de Python disponible en los nodos remotos. Por último, se copian y descomprimen los servicios de la plataforma RAI4.0 en los nodos para poder iniciar los servicios correspondientes en cada nodo.

```

1  - name: Cleanup playbook
2    hosts: aws_ec2
3    remote_user: ec2-user
4
5    tasks:
6      - include_vars: config.yaml
7
8      - name: Terminate ec2 instances
9        ec2_instance:
10          state: absent
11          region: "{{ region }}"
12          filters:
13            tag:name: "{{ instance_tag }}"
14          delegate_to: localhost
15          run_once: true
16
17      - name: Cleanup ssh config file
18        file:
19          state: absent
20          path: ssh_config
21          delegate_to: localhost
22          run_once: true

```

Script 5.4: Script Ansible para liberar todos los recursos aprovisionados mediante Ansible

En el Script 5.4 se pueden ver las dos tareas necesarias para liberar todos los recursos empleados. En primer lugar, se terminan las instancias remotas de AWS, con ello se eliminan también las unidades de almacenamiento asociadas. En segundo lugar, se elimina el fichero de configuración ssh utilizado para conectarse a los nodos remotos.

5.6.2. Bus de datos

Cuando ya se ha desplegado toda la infraestructura y se ha configurado debidamente, hay que desplegar el bus de datos. En este caso se hará empleado Kafka. Para configurar Kafka hay que seguir los siguientes pasos:

Zookeeper

En primer lugar, hay que desplegar el servicio de distribución. Para ello se establece una conexión mediante ssh a uno de los nodos de AWS y se modifica el fichero de configuración incluyendo los datos de conexión (ip, puertos, directorio de log). Tras esto se inicia el servicio y se espera a que esté disponible.

Kafka

Con el servicio de distribución activo, el siguiente paso es arrancar el servicio de mensajería. Para ello se establece una conexión a otro de los nodos aprovisionados con Ansible y se modifica el fichero de configuración con los datos de conexión a Zookeeper y las direcciones y puertos en los que se va a desplegar el servicio. Por último, se espera a que esté funcionando. En caso de querer varios brokers de Kafka para mantener la replicación, se puede repetir este proceso en diferentes nodos.

Tópicos

El siguiente y último paso es crear los tópicos en los que se publicarán los mensajes que debe procesar nuestro sistema. En este caso se han definido dos, uno donde se publicarán los datos de los sensores y otro donde se publicarán las predicciones realizadas por el sistema. Cada mensaje contendrá los siguientes campos en formato csv:

- Index: número de lectura
- Datetime: fecha hora a la que se realiza la lectura
- sensor00 - sensor51: lecturas de los sensores

Además, en el tópico de respuesta, se incluirá un campo `machine_status` que contendrá la predicción realizada por el modelo predictivo.

5.6.3. Servicio de planificación

Como servicio de planificación se utiliza Spark. Para configurarlo se necesitan al menos dos nodos: uno que haga de maestro y otro que haga de worker; aunque, en este caso, se han empleado 3: 1 maestro y 2 workers. El primer paso es conectarse al nodo que se quiere que actúe como maestro y ejecutar el servicio. Cuando el maestro esté disponible, hay que conectarse a los nodos worker y ejecutar el servicio indicando cuál es el punto de conexión con el maestro.

En este momento, el clúster ya debería estar conectado y disponible, y accediendo al puerto 8080 de la dirección del maestro desde un navegador se debería poder ver la interfaz de usuario (ver Ilustración 11) con los workers que hay disponibles y los recursos de cada uno.

The screenshot shows the Spark Master web interface at the URL `spark://ip-172-31-33-173.eu-west-1.compute.internal:7077`. It displays the following information:

- Spark Master at spark://ip-172-31-33-173.eu-west-1.compute.internal:7077**
- URL:** `spark://ip-172-31-33-173.eu-west-1.compute.internal:7077`
- Alive Workers:** 2
- Cores in use:** 8 Total, 0 Used
- Memory in use:** 26.1 GB Total, 0.0 GB Used
- Resources in use:**
 - Applications:** 0 Running, 0 Completed
 - Drivers:** 0 Running, 0 Completed
 - Status:** ALIVE

Workers (2)

Worker Id	Address	State	Cores	Memory	Resources
worker-20210429074652-172.31.40.223-40851	172.31.40.223:40851	ALIVE	4 (0 Used)	14.5 GB (0.0 GB Used)	
worker-20210429074825-172.31.38.224-39037	172.31.38.224:39037	ALIVE	4 (0 Used)	14.5 GB (0.0 GB Used)	

Running Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
----------------	------	-------	---------------------	------------------------	----------------	------	-------	----------

Ilustración 11: Interfaz de usuario de Spark con los workers y los recursos disponibles

Workflow o trabajo de Spark

El workflow o trabajo (job) como se le conoce en la nomenclatura de Spark, es el script o programa que se va a ejecutar en los nodos worker. En este caso el objetivo es que sean leídos por Spark, realice una predicción y publique el resultado en otro tópico. Para ello se ha utilizado la interfaz de programación PySpark, que permite realizar programas Spark en lenguaje Python. Concretamente, se ha usado la interfaz de Structured Streaming, que permite conectarse a una fuente de datos continua, realizar transformaciones a los datos y enviarlos a un sumidero. A continuación, se irán incluyendo fragmentos de código para comentar las diferentes fases del workflow.

En primer lugar, como se muestra en el script 5.5, hay que definir la fuente de datos a la que nos queremos conectar. En este caso, se indica que la fuente de datos será un servidor Kafka, disponible en la dirección `xxx.xxx.xxx.xxx:7077`. Se indica además el tópico al que se tiene que suscribir la aplicación, en este caso `sensor`.

```

1      # Create DataSet representing the stream of input lines
      from kafka
2      lines = spark\
3          .readStream\
4          .format("kafka")\
5          .option("kafka.bootstrap.servers", bootstrapServers)\
6          .option('subscribe', topics)\
7          .load()\
8          .selectExpr("CAST(value AS STRING)")

```

Script 5.5: Script para la conexión con la fuente de datos para iniciar el stream

El siguiente paso corresponde al procesamiento de las filas de datos que llegan de Kafka. En nuestro caso concreto los datos se presentan como una cadena de datos que contiene las lecturas de los sensores en formato csv.

```

1      # Dividir cada mensaje del topico en los diferentes
      valores
2      lecturas = lines.select(
3          split(lines.value, ',')
4      )

```

Script 5.6: Script encargado del procesamiento del stream para obtener las lecturas de bus de datos

Una vez que se tienen los datos contenidos en un dataframe (variable de datos de Spark) hay que pasárselos al modelo predictivo. Sin embargo, esto no es posible hacerlo mediante las funciones map de Spark, ya que están pensadas para trabajar únicamente con el dataframe del tipo sparkDF y el modelo desarrollado recibe como entrada un dataframe de Pandas. Por tanto, se debe implementar una función que transforme el dataframe de Spark a un dataframe de Pandas y se lo pase al modelo predictivo. Para ello se ha tenido que definir una función, para que se ejecute como cuerpo de un sumidero de tipo foreach. El propósito de estos sumideros es implementar conexiones con fuentes de datos no soportadas de forma nativa por la interfaz structured streaming de Spark. Sin embargo, en este caso, sirve para poder aplicar el modelo predictivo antes de enviar los datos a Kafka. Por tanto, como se ve en el script 5.6, se transforma la fila de datos al formato de entrada del modelo predictivo, se llama a la función predict (ver Script 5.7), y se preparan los datos para enviarlos de nuevo a Kafka a través de una función que instancia un productor.

```

1  def predict_row(row):
2      """Realiza una predicción en base a una fila de
3          lecturas de los sensores
4          <row> fila de datos del DataFrame pyspark sobre
5          la que realizar la
6          predicción.
7      """
8      # Convierte la fila en un diccionario
9      row_d = row.asDict()
10     # Transforma el diccionario en un dataframe de pandas
11     # para adecuarlo
12     # a la entrada del modelo
13     original = pd.DataFrame.from_dict(row_d, orient='index',
14         columns=col_names, dtype='float')
15     # Reemplaza los valores vacíos por np.nan para que el
16     # pipeline pueda
17     # procesarlo
18     pd_df = original.replace('', np.nan)
19     # Eliminar la columna index, ya que el modelo
20     # predictivo no la utiliza
21     pd_df.drop('index', axis=1, inplace=True)
22     # Realiza una predicción con el modelo cargado y
23     # distribuido previamente
24     prediction = broadcast_rfc_model.value.predict(pd_df)
25     # Genera un mensaje compuesto por los valores
26     # recibidos de los sensores y el resultado predicho
27     message = str(original[:,0]) + ' - ' + prediction[0]
28     # Publica el resultado en el tópic de kafka
29     kafka_sender(bytes(message.encode('utf-8')),
30         bootstrapServers, 'machine_status')
31     pass

```

Script 5.7: Función para realizar predicciones sobre cada fila de datos

Cuando está lista la función encargada de realizar las predicciones hay que definir el sumidero como se muestra en el Script 5.8. Con esto se comienza a leer y procesar los datos del tópico de Kafka y se espera a que el programa finalice. Es importante indicar que se tiene que esperar a que termine la consulta, ya que si no se hace el programa finaliza automáticamente tras el primer micro batch.

```

1      # Define un sumidero de tipo foreach para aplicar las
      transformaciones
2      # necesarias a cada registro leído del tópico de kafka y
      comienza la ejecución
3      query = lecturas.writeStream.foreach(predict_row).start()
4      # Espera a que termine la query. Esto evita que el
      programa finalice
5      # antes de terminar la consulta.
6      query.awaitTermination()

```

Script 5.8: Definición del sumidero y comienzo del streaming

Tras esto solamente falta desplegar el workflow en Spark y comenzar a enviar datos al servidor Kafka. Para lo primero se ejecuta el comando que se muestra en el script 5.9. En él se hace uso de la herramienta spark-submit proporcionada con la distribución de Spark y se le deben indicar como parámetros la dirección del nodo maestro de Spark, el fichero que contiene el programa, los paquetes necesarios y las opciones correspondientes al programa.

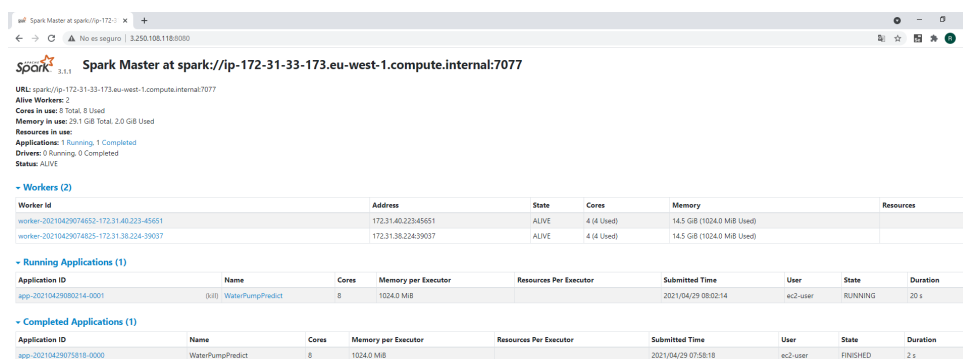
```

1 $ bin/spark-submit -master spark://<masterIP>:<masterPort>
   water-pump-predict.py \
2   --packages org.apache.spark:spark-sql-kafka-0-10_2
   .12:3.1.1 \
3   host1:port1,host2:port2 topic1,topic2 <
   rutaAlModeloPredictivo>

```

Script 5.9: Comando para iniciar la ejecución de Spark

Al ejecutar el comando se puede ver, a través de la interfaz de usuario de Spark, que el workflow está corriendo en los nodos Spark disponibles como se muestra en la Ilustración 12.



The screenshot shows the Spark Master web interface at the URL `spark://ip-172-31-33-173.eu-west-1.compute.internal:7077`. It displays the status of the Spark cluster, including the number of workers and the state of running applications.

Worker ID	Address	State	Cores	Memory	Resources
worker-20210429074652-172.31.40.223-45651	172.31.40.223:45651	ALIVE	4 (4 Used)	14.5 GB (1024.0 MB Used)	
worker-20210429074625-172.31.38.224-39037	172.31.38.224:39037	ALIVE	4 (4 Used)	14.5 GB (1024.0 MB Used)	

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210429080214-0001	WaterPumpPredict	8	1024.0 MB		2021/04/29 08:02:14	ec2-user	RUNNING	20 s

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210429075818-0000	WaterPumpPredict	8	1024.0 MB		2021/04/29 07:58:18	ec2-user	FINISHED	2 s

Ilustración 12: Interfaz de usuario de Spark con el workflow desplegado

Por último, para enviar los datos a Kafka se utiliza un software desarrollado en Java y basado en el stream ingestor disponible en [34]. Este software es capaz de leer datos de un fichero en formato csv y enviarlos con la frecuencia indicada. Para ejecutarlo usamos

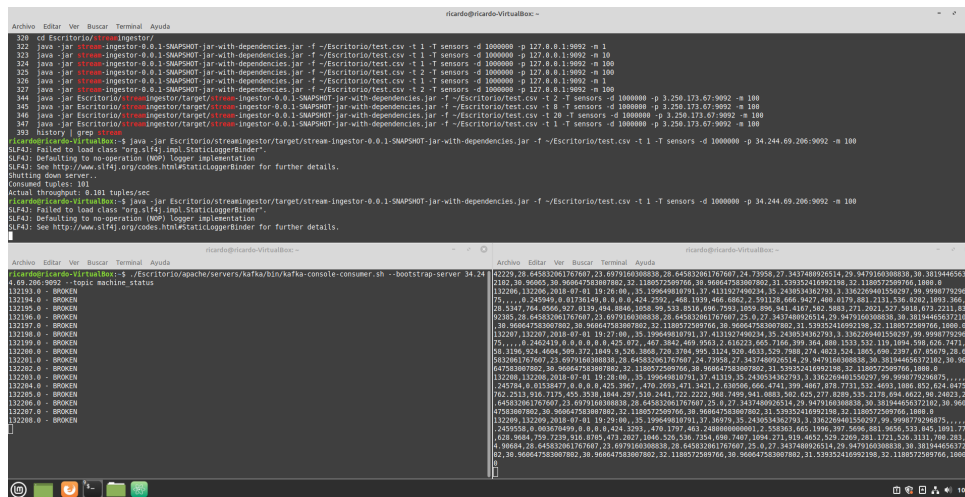
el comando del script 5.10. En este caso es necesario indicarle al programa la ruta hasta el fichero con los datos, el throughput con el que se quiere enviar los datos, el t pico, la duraci n m xima de la ejecuci n, la direcci n del servidor kafka y el n mero de l neas que se quiere leer del fichero.

```
1 $ java -jar stream-ingestor-0.0.1-SNAPSHOT-jar-with-dependencies.jar -f ../../test.csv -t 1 -T sensors -d 1000000 -p kafkaIP:PORT -m 10000
```

Script 5.10: Comando para iniciar la ejecuci n del stream-ingestor y enviar los datos a kafka

5.6.4. Ejecuci n del sistema

Tras realizar todos los pasos anteriores, el sistema deber a estar funcionando correctamente recibiendo cada instancia del t pico que se publica en kafka, proces ndola para realizar la predicci n y envi ndola de nuevo al bus de datos. Para comprobar que est  funcionando correctamente podemos crear dos consumidores por consola (uno por cada t pico). En la ilustraci n 13 se puede ver una captura de pantalla de las dos consolas en las que se muestran los datos que entran al sistema y los datos que salen. Para facilitar la interpretabilidad de la imagen y que se puedan distinguir f cilmente la entrada y la salida, en el t pico de salida solo se ha publicado el n mero de registro y el estado predicho de la m quina, aunque en la versi n final se publica junto con los valores le dos por los sensores.



Ilustraci n 13: Ejecuci n del sistema por consola

Otra parte importante del sistema es comprobar que las peticiones se atienden en el plazo esperado, para esto se puede acceder al apartado de la aplicaci n en la interfaz de usuario de Spark (ver Ilustraci n 14). En este apartado, se puede observar una l nea temporal en la que se muestra c mo son atendidas las diferentes peticiones que llegan y el tiempo requerido para ejecutar cada uno de los microbatches en los que se divide la ejecuci n.

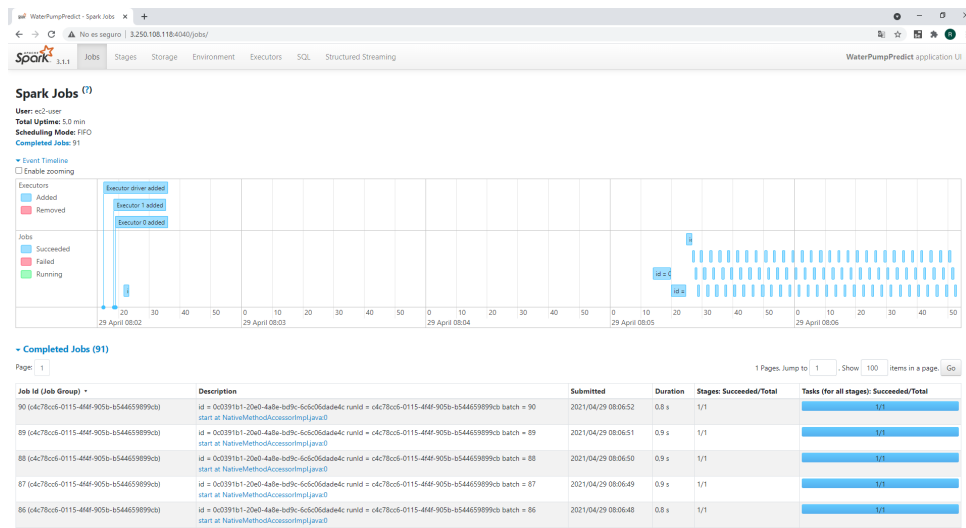


Ilustración 14: Comportamiento temporal del workflow en la interfaz de usuario de Spark

En la ilustración 14 se puede observar como cada nueva petición se envía a un nodo diferente, en este caso hay tres disponibles: el driver o nodo encargado desplegar el workflow y los dos workers definidos previamente. Esto demuestra como se aprovecha el paralelismo a la hora de realizar las predicciones, repartiendo la carga de trabajo de forma equitativa entre todos los nodos disponibles, otorgando mayor flexibilidad a la hora de escalar el sistema, ya que se puede hacer de forma vertical (aumentando la potencia de los nodos existentes) y horizontal (aumentando el número de workers registrados en el sistema). Estos valores se almacenan en un sistema de log, por lo que se podrían emplear para, junto a tecnologías como Ansible o AWS Cloudwatch, para implementar un sistema de notificaciones cuando disminuya el rendimiento o incluso automatizar el escalado del sistema.

5.7. Interfaz de usuario

Para demostrar una de las posibles aplicaciones de este sistema de mantenimiento se ha desarrollado una sencilla interfaz de usuario en la que se puede monitorizar en tiempo real cada uno de los sensores y el estado de la máquina. Esta interfaz de usuario es solo un ejemplo, pero cualquier aplicación que forme parte del ecosistema y se conecte al bus de datos es susceptible de aprovechar este tipo de tecnología. En el caso de un sistema en funcionamiento se podría reemplazar esta interfaz por una herramienta de monitorización existente.

La interfaz cuenta principalmente con dos módulos: un servicio encargado de recibir los datos publicados en Kafka y enviarlos a través de un websocket, y una aplicación web que se conecta al websocket y presenta la información en unas gráficas que se actualizan en tiempo real.

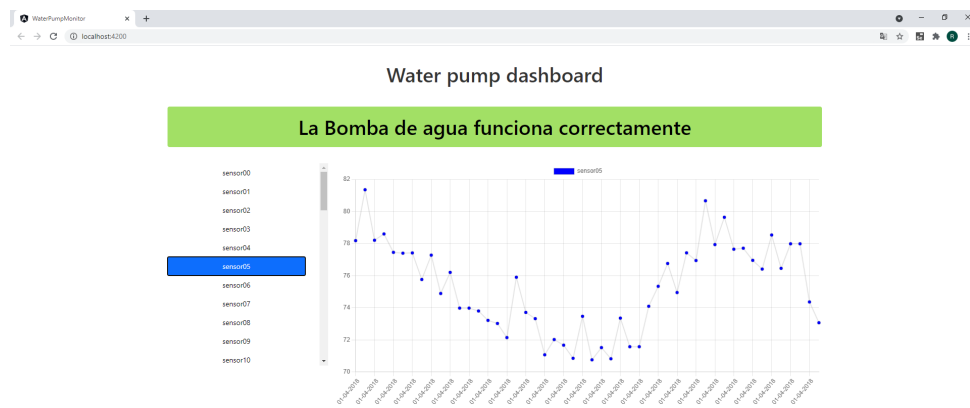
5.7.1. WebSocket

Este componente de la interfaz está desarrollado con el framework Spring boot y tiene como misión establecer una conexión con Kafka y hacer los datos accesibles para la capa de presentación a través de un puerto concreto.

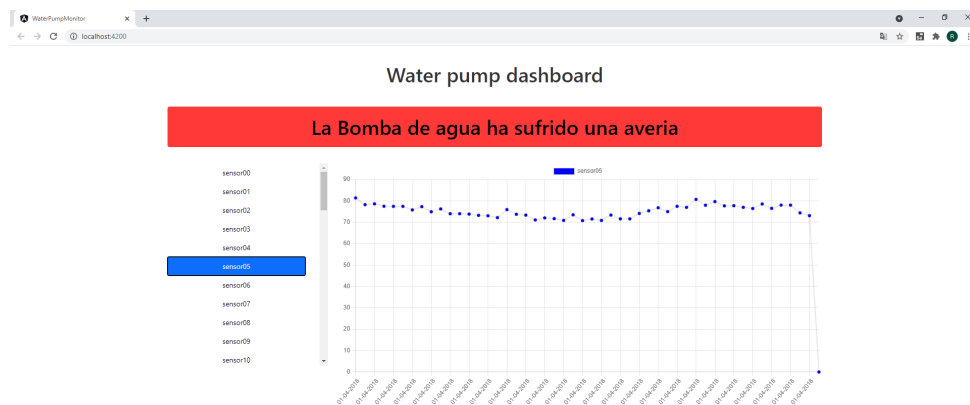
5.7.2. Capa de presentación

Este componente se ha desarrollado utilizando Angular y typescript como lenguaje de programación. La capa de presentación cuenta con un service (elemento de angular destinado al acceso a datos) a través del que se obtienen los datos del websocket y un component (elemento de diseño personalizado) en el que se estructura la página web. Este component recibe una instancia del service y se suscribe a él para actualizar las gráficas cada vez que se obtenga un dato nuevo.

En la página web (ver Ilustración 15) se puede ver la evolución de los valores a través de unas gráficas desarrolladas usando el paquete charts.js y un cartel de color verde (Ilustración 15(a)) o rojo (Ilustración 15(b)) que indica el estado de la bomba de agua.



(a) Funcionamiento correcto



(b) Error de funcionamiento

Ilustración 15: Interfaz de usuario desarrollada con Angular

6 Aportaciones metodológicas y consideraciones para el análisis de flujos de datos

En este apartado se señalan qué tareas deben incluirse o aquellas que requieren especial atención en el análisis de flujos de datos. Como se ha visto en el caso de estudio, no es necesario introducir ni eliminar fases de la metodología CRISP-DM, sin embargo si que es necesario tener en cuenta algunos factores a la hora de abordar algunas de ellas.

En la fase de entendimiento del problema, hay que prestar especial atención al tipo de fuente que genera los datos, ya que no es lo mismo que los datos sean generados por el comportamiento humano que por una máquina. En el primero de los casos, tendremos que tener muy presente el concept drift y adaptar todo el proceso de entendimiento del conjunto de datos y modelado para emplear técnicas capaces de adaptarse a los cambios y desechar los datos obsoletos. Mientras que en el segundo caso no será necesario, ya que el comportamiento de la máquina debería ser constante a lo largo del tiempo cuando opera normalmente.

En la fase de entendimiento del conjunto de datos, habrá que tener especial cuidado al escoger el conjunto de datos a analizar, ya que el volumen de datos será muy grande y la muestra debe ser suficientemente representativa para permitir al algoritmo detectar la ventana temporal en la que se producen los cambios de tendencia (en el caso de que se aplique el concept drift), así como disponer un subconjunto equilibrado de tuplas de cada clase (en el caso de la clasificación).

En la fase de preparación de los datos es importante optar por estrategias que sean eficientes en tiempo, ya que la velocidad a la que se generarán los datos en el sistema final va a suponer que una limitación a la hora de procesar los datos en tiempo real. Por tanto, es preferible optar por estrategias paralelizables, que no conlleven realizar demasiados cálculos, así como aplicar técnicas de reducción de la dimensionalidad.

En la fase de modelado se debe tener presente que se manejan cantidades de datos elevadas y, por tanto, los tiempos de entrenamiento de los modelos también lo serán. Esto hace que sea muy recomendable utilizar modelos sencillos o que se puedan aprovechar del paralelismo para el proceso de entrenamiento. Además, para tratar con el concept drift es necesario que el modelo sea capaz de extraer conocimiento de los nuevos datos de manera dinámica para adaptarse a los cambios de tendencia, ya que en muchas ocasiones no se podrán almacenar los datos debido al gran volumen o incluso a las restricciones de privacidad. Asimismo, para conseguir que el modelo sea más robusto y sea capaz de responder ante un mayor número de situaciones, se puede optar por una estrategia de voting. Como se ha demostrado en este proyecto, a pesar de conseguir unos resultados muy buenos con los modelos simples, que la estrategia de voting ha conseguido mejorarlos ligeramente. Por otra parte, conviene asegurarse que los modelos soporten algún meca-

nismo de serialización, ya que al tratarse de un sistema distribuido es probable que sea necesario transmitirlos a través de la red al arrancar el sistema.

En la fase de evaluación se debe tener en cuenta, no solo las métricas escogidas para medir la calidad de las predicciones, si no también el tiempo requerido para realizar una predicción, ya que esto va a determinar que se pueda cumplir con las restricciones temporales del sistema. Se ha de señalar que, esta fase debe desarrollarse en conjunto con la fase de despliegue, ya que las latencias y el comportamiento ante los nuevos datos son determinantes a la hora de evaluar la validez del modelo. Cuando nos encontramos ante un conjunto de datos estático, con un número de instancias pequeño, o que no se vea afectado por el concept drift o las restricciones temporales propias de la industria, la fase de evaluación se puede realizar de forma aislada.

A continuación, se señalan retos aún abiertos en este ámbito. En relación al análisis de grandes volúmenes de datos Sivarajah et al. [1] los clasifican en las tres categorías que se muestran en la Ilustración 16. Las consideraciones y modificaciones de CRISP-DM mencionadas en este apartado responden principalmente a las dos primeras categorías, esto es, los retos relativos a los datos y los retos relativos al proceso, sin embargo, quedan por tratar los retos relativos a la gestión de los datos. Este ámbito, es posiblemente el más complejo de todos y es por eso que para la construcción y el despliegue del sistema conviene disponer de una plataforma tecnológica que siga una arquitectura como la propuesta en RAI4.0, la cual integra los servicios de gobierno y seguridad de los datos lo que permite cubrir esa faceta.

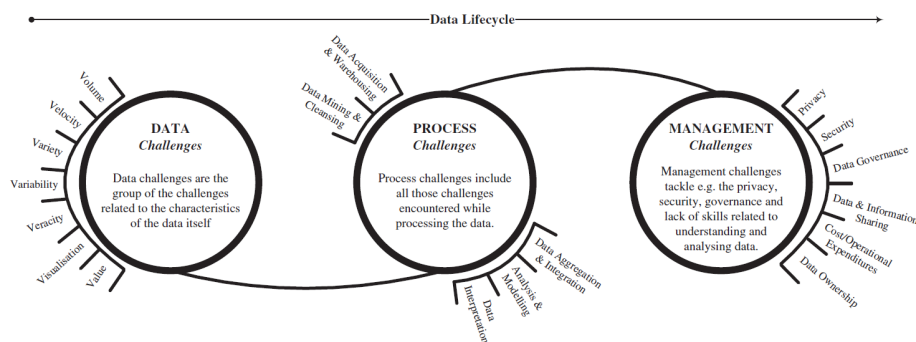


Fig. 1. Conceptual classification of BD challenges.

Ilustración 16: Clasificación de retos del análisis de Big Data[1]

Por otra parte, y en relación al ámbito big data, y en particular al análisis de datos, Abedallah Zaid Abualkishik [35] señala que los resultados basados en el análisis estadístico no se deben tratar de la misma forma que en la analítica de datos tradicional, como consecuencia del gran volumen de datos y la acumulación de error que puede ocurrir en el pipeline. Además, menciona que no todas las técnicas son aplicables al análisis de big data debido a la limitación en la escalabilidad, la heterogeneidad de los datos y la correlación espuria. Por ello se hace necesario desarrollar nuevas técnicas de modelado y tratar de mejorar o adaptar las existentes para aumentar su escalabilidad y desempeño con el big data.

7 Conclusiones y líneas futuras

La Industria 4.0, también llamada industria inteligente, se considera la cuarta revolución industrial y busca transformar a la empresa en una organización inteligente para conseguir los mejores resultados de negocio. Entre otras tecnologías habilitadoras está la IA y el machine learning aplicado a la ingente cantidad de datos que el entorno genera [2].

El objeto de este trabajo ha sido el desarrollo y despliegue en la nube AWS de un sistema de mantenimiento predictivo construido por medio de técnicas de minería de datos sobre un flujo continuo de datos procedente de la sensórica ubicada en una bomba que proporciona agua a una pequeña localidad. Este sistema big data se ha diseñado bajo una arquitectura distribuida y orientada al dato denominada RAI4.0 [3].

Para su consecución se ha escogido un conjunto de datos que represente un flujo continuo de datos similar a lo que se espera encontrar en un entorno industrial. En este caso, corresponden a las mediciones recogidas por un conjunto de sensores instalados en una bomba de agua. Sobre este conjunto de datos se realizó un proceso de minería de datos siguiendo la metodología CRISP-DM para construir una serie de predictores y desplegarlos en una plataforma distribuida alojada en la nube utilizando tecnologías Apache (Kafka para el bus de datos, Zookeeper como coordinador y Spark como planificador). El despliegue en AWS se realizó con Ansible. Por otra parte, con objeto de visualizar el estado de la bomba y su predicción, se construyó una sencilla interfaz gráfica que permite visualizar en tiempo real los datos generados por el sistema de mantenimiento predictivo. Esto no es más que una manera de explotar esta información, aunque se podría utilizar la salida del predictor como entrada para cualquier otro *workflow* o proceso que se despliegue en el entorno.

Adicionalmente, en el capítulo 6, se ofrecen pautas para la implantación de este tipo de soluciones IA en la industria y se señalan retos aún abiertos en el campo.

Como líneas de trabajo futuras de este proyecto se deberían afrontar casos de uso en el que tenga importancia el *concept drift*. Por ejemplo un sistema recomendador permita la producción a la demanda en un sistema de fabricación. Esto supone aplicar otro tipo de técnicas y trabajar con el concepto de ventana temporal [36].

Otro aspecto a tener en cuenta en el futuro, es que este proyecto se ha realizado con un conjunto de datos ya recolectados y se ha simulado el flujo continuo de datos leyendo de un fichero csv. Sin embargo, en un caso real hay que abordar la recolección de los datos implementando algún mecanismo de sincronización entre los sensores, que publicarán los datos de forma independiente en el bus de datos. Para ello será necesario, construir un experimento en el que se cuente con varios sensores reales.

En cuanto al proceso de construcción de modelos, cabe destacar que el entrenamiento de los modelos se ha llevado a cabo en una sola máquina y, por tanto, cuando los modelos eran algo más complejos y el número de parámetros de entrada era alto, el tiempo de entrenamiento aumentaba notablemente. La única forma de acelerar el proceso de entre-

namiento y ajuste de parámetros durante el desarrollo de este proyecto es utilizar una máquina más potente. Esta estrategia no es práctica, ya que se necesita una máquina con una capacidad de procesamiento desproporcionada con las demandas del sistema solamente para llevar a cabo la fase de entrenamiento. Por esto, es interesante encontrar o en su caso, desarrollar técnicas que permitan abordar el entrenamiento y ajuste de los modelos de forma distribuida.

Actualmente el sistema es capaz de detectar los fallos que están ocurriendo en la máquina en el instante en el que se han tomado las medidas, pero no es capaz de saber cuándo se va a estropear. Para conseguir un mayor aprovechamiento de esta plataforma, sería interesante construir otro tipo de modelos predictivos capaces de predecir un fallo con algunas horas de antelación o incluso con un día de antelación.

8 Referencias

- [1] U. Sivarajah, M. M. Kamal, Z. Irani, and V. Weerakkody, “Critical analysis of big data challenges and analytical methods,” *Journal of Business Research*, vol. 70, pp. 263–286, 2017.
- [2] R. Y. Zhong, X. Xu, E. Klotz, and S. T. Newman, “Intelligent manufacturing in the context of industry 4.0: A review,” *Engineering*, vol. 3, no. 5, pp. 616–630, 2017.
- [3] R. D. Herrero, P. L. Martínez, and M. Zorrilla, “Arquitectura de referencia para el diseño y desarrollo de aplicaciones para la industria 4.0,” *Revista Iberoamericana de Automática e Informática industrial*, vol. 0, no. 0, 2021.
- [4] J. Gama, *Knowledge Discovery from Data Streams*. Chapman Hall/CRC, 1st ed., 2010.
- [5] H. Li, D. Parikh, Q. He, B. Qian, Z. Li, D. Fang, and A. Hampapur, “Improving rail network velocity: A machine learning approach to predictive maintenance,” *Transportation Research Part C: Emerging Technologies*, vol. 45, pp. 17 – 26, 2014. Advances in Computing and Communications and their Impact on Transportation Science and Technologies.
- [6] T. Praveenkumar, M. Saimurugan, P. Krishnakumar, and K. Ramachandran, “Fault diagnosis of automobile gearbox based on machine learning techniques,” *Procedia Engineering*, vol. 97, pp. 2092 – 2098, 2014. "12th Global Congress on Manufacturing and Management" GCMM - 2014.
- [7] R. Prytz, S. Nowaczyk, T. Rögnvaldsson, and S. Byttner, “Predicting the need for vehicle compressor repairs using maintenance records and logged vehicle data,” *Engineering Applications of Artificial Intelligence*, vol. 41, pp. 139 – 150, 2015.
- [8] M. Canizo, E. Onieva, A. Conde, S. Charramendieta, and S. Trujillo, “Real-time predictive maintenance for wind turbines using big data frameworks,” in *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 70–77, 2017.
- [9] Oracle, “¿Qué es Java y para qué es necesario?.” https://www.java.com/es/download/help/whatis_java.html, n.d.
- [10] Python Software Foundation, “About python | python.org.” <https://www.python.org/about/>, 2021.
- [11] Wikipedia, “TypeScript — Wikipedia, the free encyclopedia.” <http://es.wikipedia.org/w/index.php?title=TypeScript&oldid=133843781>, 2021. [Online; accessed 06-May-2021].

- [12] “Spring | Home.” <https://spring.io/>.
- [13] Wikipedia, “Spring Framework — Wikipedia, the free encyclopedia.” <http://es.wikipedia.org/w/index.php?title=Spring%20Framework&oldid=135276118>, 2021. [Online; accessed 06-May-2021].
- [14] Wikipedia, “Angular (framework) — Wikipedia, the free encyclopedia.” [http://es.wikipedia.org/w/index.php?title=Angular%20\(framework\)&oldid=131996580](http://es.wikipedia.org/w/index.php?title=Angular%20(framework)&oldid=131996580), 2021. [Online; accessed 06-May-2021].
- [15] The Apache Software Foundation., “Apache kafka.” <https://kafka.apache.org/>, 2017.
- [16] The Apache Software Foundation., “Apache zookeeper.” <https://zookeeper.apache.org/>, 2017.
- [17] The Apache Software Foundation., “Apache spark - unified analytics engine for big data.” <https://keras.io>, 2017.
- [18] F. Chollet *et al.*, “Keras.” <https://keras.io>, 2015.
- [19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [20] J. D. Hunter, “Matplotlib: A 2d graphics environment,” *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.
- [21] Wes McKinney, “Data Structures for Statistical Computing in Python,” in *Proceedings of the 9th Python in Science Conference* (Stéfan van der Walt and Jarrod Millman, eds.), pp. 56 – 61, 2010.
- [22] Amazon Web Services, “Aws | cloud computing - servicios de informática en la nube.” <https://aws.amazon.com/es/>, n.d.
- [23] Amazon Web Services, “Aws | elastic compute cloud (ec2) de capacidad modificable en la nube.” <https://aws.amazon.com/es/ec2/>, n.d.
- [24] Amazon Web Services, “Aws | elastic block store (ebs) para almacenamiento persistente.” <https://aws.amazon.com/es/ebs>, n.d.
- [25] Amazon Web Services, “Aws | almacenamiento de datos seguro en la nube (s3).” <https://keras.io>, n.d.
- [26] Red Hat, “Apache spark - unified analytics engine for big data.” <https://www.ansible.com/>, n.d.
- [27] R. D. Herrero, “DintenR/TFM-Data-Stream-Mining.” <https://github.com/DintenR/TFM-Data-Stream-Mining>, 2019.
- [28] C. Shearer, “The crisp-dm model: the new blueprint for data mining,” *Journal of data warehousing*, vol. 5, no. 4, pp. 13–22, 2000.
- [29] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, “From data mining to knowledge discovery in databases,” *AI magazine*, vol. 17, no. 3, p. 37, 1996.

- [30] SAS, “SAS help center: Introduction to SEMMA.” <https://documentation.sas.com/?docsetId=emref&docsetTarget=n061bzurmej4j3n1jnj8bbjjm1a2.htm&docsetVersion=14.3&locale=en>, 2017.
- [31] M. R. Berthold, C. Borgelt, F. Hppner, and F. Klawonn, *Guide to Intelligent Data Analysis: How to Intelligently Make Sense of Real Data*. Springer Publishing Company, Incorporated, 1st ed., 2010.
- [32] UnknownClass, “pump_sensor_data | kaggle.” <https://www.kaggle.com/nphantawee/pump-sensor-data>, 2019.
- [33] R. Kohavi, “A study of cross-validation and bootstrap for accuracy estimation and model selection,” vol. 14, 03 2001.
- [34] M. Algorri, “Github - algorri94/streamingestor: A simple tool for streaming a file to a kafka cluster.” <https://github.com/algorri94/streamingestor>, 2016.
- [35] A. Abualkishik, “Hadoop and big data challenges,” *Journal of Theoretical and Applied Information Technology*, vol. 97, p. 3488, 06 2019.
- [36] S. Wares, J. Isaacs, and E. Elyan, “Data stream mining: methods and challenges for handling concept drift,” *SN Applied Sciences*, vol. 1, p. 1412, Oct 2019.